Management of the Internet and Complex Services

European Sixth Framework Network of Excellence FP6-2004-IST-026854-NoE

Deliverable 8.3
Definition of a Draft Extended IP Network Management Model

The EMANICS Consortium

Caisse des Dépôts et Consignations, CDC, France
Institut National de Recherche en Informatique et Automatique, INRIA, France
University of Twente, UT, The Netherlands
Imperial College, IC, UK
Jacobs University Bremen, JUB, Germany
KTH Royal Institute of Technology, KTH, Sweden
Oslo University College, HIO, Norway
Universidat Politecnica de Catalunya, UPC, Spain
University of Federal Armed Forces Munich, CETIM/UniBwM, Germany
Poznan Supercomputing and Networking Center, PSNC, Poland
University of Zürich, UniZH, Switzerland
Ludwig-Maximilian University Munich, LMU, Germany
University of Surrey, UniS, UK
University of Pitesti, UPI, Romania

For more information on this document or the EMANICS project, please contact:

Dr. Olivier Festor
Technopole de Nancy-Brabois — Campus scientifique
615, rue de Jardin Botanique — B.P. 101
F—54600 Villers Les Nancy Cedex
France

Phone: +33 383 59 30 66
Fax: +33 383 41 30 79
E-mail: <olivier.festor@loria.fr>

Document Control

Title: Definition of a Draft Extended IP Network Management Model

Type: Public

Editors: Martin Waldburger, Burkhard Stiller

E-mail: waldburger@ifi.uzh.ch, stiller@ifi.uzh.ch

Authors: Mark Burgess, Frank Eyermann, Stylianos Georgoulas, Hasan, Thomas Schaaf, Joan Serrat, Burkhard Stiller, Martin Waldburger (alphabetic order)

Doc ID: D8.3-v1.0

AMENDMENT HISTORY

Version	Date	Author	Description/Comments
V0.1	December 3, 2007	Burkhard Stiller (BS)	Document structure, ToC, Intro, Distribution of responsibilities
V0.2	December 4, 2007	Martin Waldburger (MW), Hasan (HH), Cristian Morariu (CM)	GridAcc, ASAM, 2 Papers
V0.3	December 14, 2007	HH, Stylianos Georgoulas (SG), Frank Eyermann (FE), MW, Thomas Schaaf (TS), Joan Serrat (JS)	ASAM update, AURIC paper, GridAcc update, BP3EM update, PRIPOL update
V0.4	December 16, 2007	BS	Intro, Exec Summary, editing
V0.5	December 17, 2007	BS	Complete re-read, inclusion of TODO items, updates across the document
V0.6	December 19, 2007	MW, HH, FR, SG	Update of respective sections
V0.7	December 21, 2007	BS, Mark Burgess (MB), MW, JS, TS	Input consolidated, edited input from MB, editing comments removed, annex prepared, edited input from JS, edited input from TS
V0.8	December 28, 2007	BS	Final editing and open issues determined, comments addressed to respective partners
V0.9	January 12, 2008	BS, MW, TS	Update of final partner input
V1.0	January 16, 2008	BS	Editing of final input from GA/WP8 Meeting in Barcelona, ready for submission to Commission. Attachment of papers.

Legal Notices
The information in this document is subject to change without notice.
The Members of the EMANICS Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the EMANICS Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Table of Content

1 Executive Summary

Managing the Internet as well as its related services and traffic determines the important goal for any operational network. This becomes even more obvious as commercial interests in service provisioning and use will move to the foreground. Thus, EMANICS in general and its Work Package 8 (WP8) on Economic Management specifically intends to understand, to determine interrelations, and to simplify such Internet-based network management tasks.

In continuation of the previous project year, this area of work has been addressed in the at hand Deliverable D8.3 explicitly under an integrated point of view of three major methodological points of view, determining the scope:

1 the multi-domain aspect,

2 the optimization of existing IT service management approaches, and

3 concrete economic or economics-supporting mechanisms.

While specifically topic 1 and 3 are combined to the multi-domain auditing in the ASAM approach, the Service Level Agreement (SLA) application on Grid services within the context of promise theory address topic 2 in the SaPDoGS approach. Furthermore, the BP3EM approach outlines key gaps between IT service management and system administration, which are utilized to develop the Service Monitoring Architecture (SMONA), all of which addresses topic 2. Additionally, the pricing by policies approaches offers an alternative path to determine the market price of services based on policies, addressing topic 3. Finally, the combination of topic 2 and 3 is addressed by the Grid accounting approach, which combines the concrete accounting mechanism for Grids as an optimization for computing centers and their cost structure. Therefore, Figure 1 outlines these dependencies determined and investigated in more detail throughout D8.3.

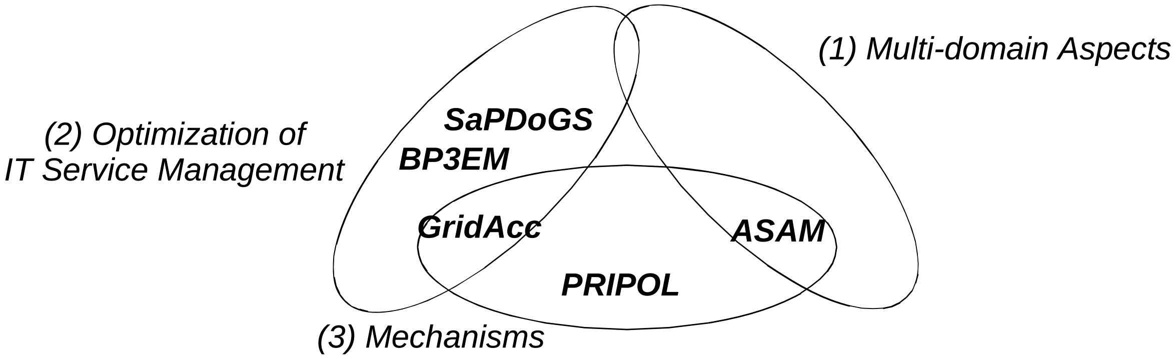


Figure 1: Scope of D8.3 Dealt by within 5 EMANICS WP8 Approaches

The key results obtained in D8.3 of WP8 in EMANICS are determined by the fact that the extension of a network management model with auditing is feasible in the multi-domain case sketched. Further performance investigations will follow toward the end of this project. SaPDoGS proposes to construct a model of Grid services by documenting necessary and sufficient promises to enable the collaboration between Grid nodes. The Service Monitoring Architecture has been defined and the handbook on integrating cfengine with ITIL processes as an example on how to merge business-driven IT Management and traditional systems administration is under way. The pricing by policies work combines the traffic estimation matrix with the resource availability matrix by means of traffic engineering and service management mechanisms. Finally, the applicability of the resource-based, highly flexible accounting model for dynamic Virtual Organizations shows that combining both, technical and economic accounting by means of Activity-based Costing (ABC) service constituent parts and defined accountable units is highly valid.

A special note and success has to be named explicitly: The two papers written in the context of EMANICS'' WP8 work in Section 13.2 has been awarded the ***"Best Student Paper Award"***: C. Morariu, M. Feier, B. Stiller: LINUBIA: A Linux-supported User-Based IP Accounting. Lecture Notes in Computer Science, Vol. 4785, Springer 2007, pages 229-241, 18th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2007), San Jose, California, U.S.A., October 2007. Additionally, the paper in Section 13.4 has been awarded the ***"Best Paper Award"***: 2nd International Conference on Internet Monitoring and Protection (ICIMP 2007), Silicon Valley, U.S.A., July 1-6, 2007, Organized by International Academy, Research, and Industry Association (IARIA).

2 Introduction

Managing the infrastructure of tomorrow's Internet requires suitable technology and mechanisms as well as valid and viable economic means, which will lead to a generic information infrastructure and which enables multiple parties' access — even between *multiple domains* — in an economically fair manner. In particular the economic dimension of network management for Internet Service Providers (ISP) has to be an integrated part of an overall IP-based network solution.

While different measures depend on various factors and influence providers’ models, their *mechanisms* for *accounting*, traffic control, *auditing*, network state supervision, their management optimization dimension, and their way of technical operations, *e.g.*, by admission control schemes as well as by *Service Level Agreements (SLA) management schemes*, have to be integrated for a suitable economic management model.

While major factors have been addressed until now in the research community as separated activities, such as achieving technical efficiency of accounting modules, definitions of charging models independent of technological prerequisites, maintaining viable Service Level Management (SLM) aspects in *IT service management*, operating admission control, and determining management-independent services models, the core inter-operation model of the most essential of those components has been determined in Phase I of EMANICS.

On this basis, the development of a suitable ***service management architecture*** and corresponding ***selected functionality*** as a foundation for the integration for a production environment has to be refined in Phase II, mainly with respect to efficient interfaces, operationally applicable mechanisms, Value-added Service provisioning, and based on a heterogeneous technology infrastructure. In addition, advanced traffic management mechanisms, being dependent on economic measures, such as *pricing*, as well as auditing will be considered to enhance the QoS instrumentation and compliance checking for IP-based services.

In consequence, the envisaged future IP networks will have the potential to become enablers for attractive new kinds of services and applications in the business and private sectors, such as Grid services. Therefore, the need for appealing new service offerings is emerging, and which are provided to customers with a guaranteed QoS at an agreed upon cost. The SLA does form in all of these cases the contractual obligation to be fulfilled and negotiated.

Therefore, the selected focus of Work Package 8 work has been lead on five different key aspects of this ***extended IP network management model***. Firstly, the *multi-domain auditing* approach (ASAM, cf. Table 1) addresses the domain-embracing aspect of network management functionality in the case of the business-relevant Service Level Objective (SLO) auditing mechanism.

Secondly, the application of promise theory onto Service Level Agreements — in the specialized application domain of Grid services — is addressed in the SaPDoGS approach (cf. Table 1), which determines a new *SLA management scheme* being part of the network management model.

Thirdly, the strong aspects of IT service management applied to system administration are relevant to enable *management optimizations of existing management mechanisms*. Thus, best practices, processes, and promises in economic management are handled in the BP3EM approach (cf. Table 1).

Fourthly, pricing determines the traditional *economic mechanism* to determine an applicable price for a service in use. The PRIPOL approach (cf. Table 1) follows this goal by searching for new and *advanced ways to determine the market price based on policies*.

Finally, the Grid service accounting approach GridAcc (cf. Table 1) deals with the concrete application of WP8 Phase I results in Phase II, in which the *accounting model for Grid services* is evaluated in a real computing center, showing the key model-based cost calculations for the set of Grid services in place.

Table 1: Work Package 8 (WP8) Projects, Their Durations, and Partners Involved

Acronym	Full Project Title	Start Time	End Time	Partners
ASAM	Auditing of SLOs Across Multiple Provider Domains	M+19	M+36	UniBwM, UniZH, UniS
SaPDoGS	SLA and Promise Descriptions of GRID Services	M+19	M+36	HIO, UniBwM
BP3EM	Best Practices, Processes and Promises in Economic Management	M+19	M+36	LMU, HIO
PRIPOL	Pricing by Policies	M+19	M+36	UPC, KTH, UniZH
GridAcc	Grid Accounting	M+19	M+30	UniZH, UniBwM, LMU/LRZ

2.1 Document Outline

This deliverable D8.3 is organizes as follows, which is in-line with the Phase II project-based operation of EMANICS. Driven by the 5 projects within Work Package 8, namely ASAM (Auditing of SLOs Across Multiple Provider Domains), SaPDoGS (SLA and Promise Descriptions of GRID Services), BP3EM (Best Practices, Processes and Promises in Economic Management), PRIPOL (Pricing by Policies), and GridAcc (Grid Accounting) the following chapters are devoted to each of them and the current state achieved.

Due to the fact that each project has a different run-time and effort, those chapters differ very much in length and depth. Thus, the final result and status of a full-fledged design, evaluation, and prototypical implementation and modeling respectively will become part of Deliverable D8.4 to be expected by end of 2008.

Following these introductory remarks and document overview in Section 2, ASAM's status is presented in Section 3, which does include an initial systems design for multi-domain auditing functionality. Section 4 covers in an abstract type of manner the goals and scope of the SaPDoGS project. Additionally, the business process-related point of view of IT service management is addressed in Section 5, where BP3EM outlines gaps between IT service management and system administration, which are utilized to develop the Service Monitoring Architecture (SMONA).

Furthermore, the PRIPOL project considering the problem of getting a market price for a service as a problem that can be formulated in terms of policies, which have to be deduced from high level service provider objectives, is outlined in Section 6. Technologically, Section 7's work on Grid Accounting — originating from WP8 Phase II — was driven by the successful preliminary functional evaluation of the accounting model developed. Thus, this accounting model now undergoes a full-fledged evaluation based on the specific use case considerations with the Leibniz Supercomputing Centre in Germany. Finally, Section 8 draws preliminary conclusions of these 5 projects undertaken in Work Package 8 and outlines the next steps in WP8.

3 ASAM: Auditing of SLOs Across Multiple Provider Domains

Today, a user normally concludes a Service Level Agreement (SLA) with a particular network provider and there are SLAs established among several providers. Considering a scenario, in which users require to know the end-to-end performance of a transport service that their application is using across several network providers of a virtual organization, automated auditing of the performance of each network provider to meet the overall performance requirement related to this specific user is crucial. Thus, the main objective of this work is to develop an architecture and a new protocol, if required, for metering and auditing of network performance across multiple, co-operating network providers. In this context, the auditing of SLAs defines the process of monitoring of whether a service provider delivers agreed upon service levels or not. While frameworks exist to monitor application-level SLAs, the end-to-end monitoring of IP-carrying SLAs, especially in a multi-domain environment like the Internet, is still an open issue.

In previous work, requirements towards auditing of IP-carrying SLAs were analyzed and a protocol and architecture for an auditing system was sketched. Unlike existing approaches, in which a provider proves compliance of its service delivery according to Service Level Objectives (SLOs) valid for all customers within its own domain, and publishes the results, this work proposes metering and auditing to be initiated on demand by authorized users of those network services invoked. Special interest will be taken in (a) creating Internet-like simulation scenarios (extension for ns-2) for researching and validating monitoring and measurement approaches and (b) developing a prototype for applying such mechanisms in test-beds.

3.1 Introduction

Users of Internet services are obviously interested in end-to-end performance of a service, which often involves data transfer across several networks owned by different providers. However, a provider is usually only willing to define Service Level Objectives which only cover performance commitments within its own domain. This situation is frustrating since users are not satisfied with SLOs defined and providers seem to be *unable*, not only *unwilling*, to give commitments on end-to-end service performance guarantees. Thus, it is necessary to design and provide an operational environment where a provider will not only be willing to, but will also actually be able to offer end-to-end performance guarantees across multiple domains.

Clearly, this requires providers to cooperate in order for them to offer end-to-end performance guarantees. To attract a provider to cooperate, such a co-operation should be or at least have the potential to be beneficial for each of the cooperating members. This is a challenge for cooperation designers. At the same time, services offered today are becoming more complex. Fortunately, through the Web and Grid technology, a service can be composed of other more simpler services. This leads to service abstraction, and since composed services may comprise services from various providers, this also leads to organization abstraction. Abstraction defines the technique for hiding underlying mechanisms. A Virtual Organization (VO) [6], [7] is a form of organization abstraction. It is understood as a temporary or permanent coalition of geographically dispersed individuals, groups, organizational units or entire organizations that share resources, capabilities and information to achieve common objectives. VOs can provide services and thus, can take the role of a service provider.

3.2 System Design

Figure 3 depicts an overview of the ASAM architecture which consists of three types of networks: core networks, access networks, and access client networks. Core networks are responsible for efficient transfer of network packets, while access networks provide for connectivity service to their clients, which are located in access client networks. Access client networks contain servers for value added services (including applications beyond network services) and end systems (ES) as clients of those value added services. Depending on the type of network a provider is operating, ASAM distinguishes between backbone network providers, access network providers, and value added service providers. These different types of providers can be seen as different provider roles. Furthermore, there is a role termed end-customer which does not provide anything but only consumes services. However, end-customers may also interact or communicate with each other using applications on end systems as well as using network services for data transfer.

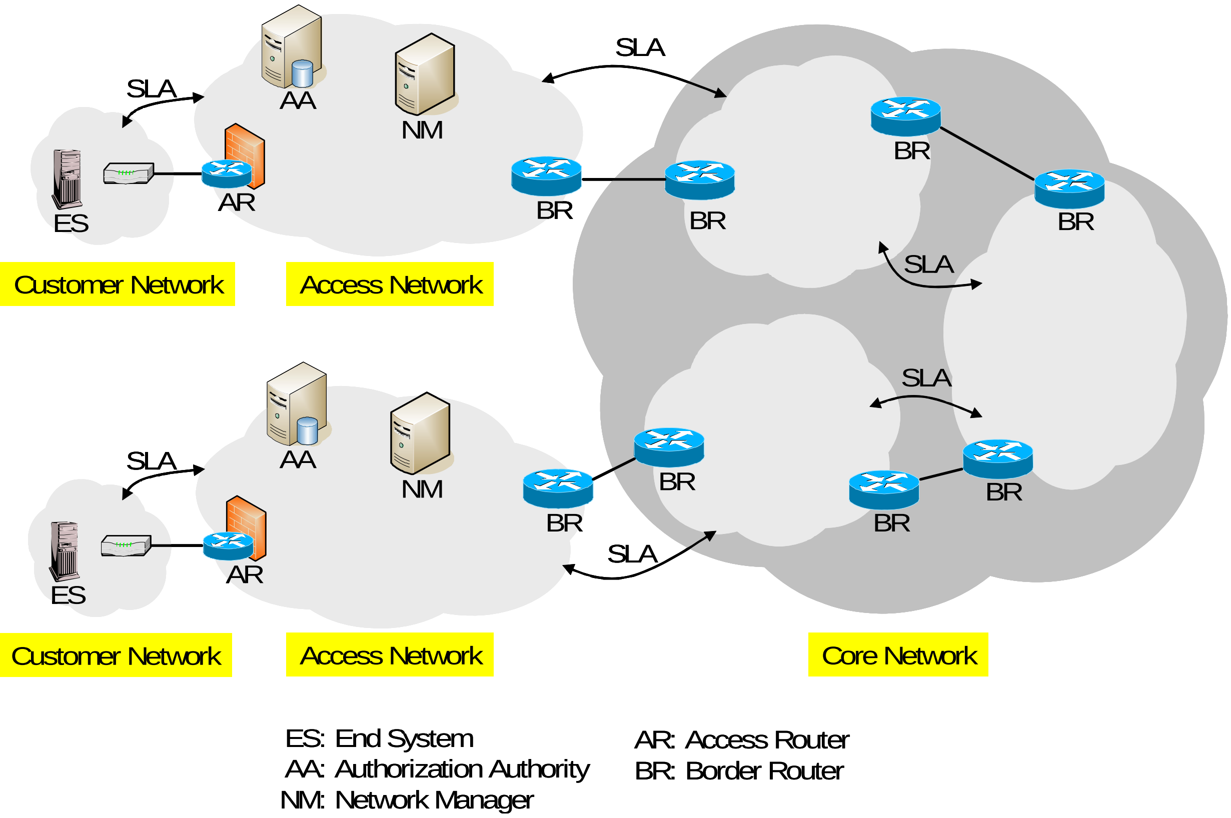


Figure 2: ASAM Architecture Overview

In principle, a party can own several networks of the same or different types, and therefore can take one or more roles. An SLA is established between two parties, *not* between two roles. However, considering the finest role assignment possible, each party is assumed to have one single role. Thus, an end-customer is supposed to conclude an SLA with an access network provider in order to be able to connect to the Internet and to consume value added services. This is also true for a value added service provider in order to enable it to deliver services. Furthermore, network providers are assumed to establish SLAs or peering agreements among themselves.

To allow for offering end-to-end performance guarantees across multiple provider domains, a VO needs to be established, which comprises all network providers and value added service providers along the data path to be taken by the services used. This VO will then be able to offer end-to-end performance guarantees to end-customers. There are some benefits for a provider to be part of a VO. From the end-customer point of view the end-to-end performance is important. Thus, it is irrelevant for the end-customer if some of the providers within the VO cannot meet their individual SLOs, as long as the end-to-end SLOs are held by the VO. This means, a provider may benefit from a performance increase of other providers to temporarily compensate for its own performance decrease.

However, if the VO is not able to meet the end-to-end performance guarantees, it is necessary to know the domains in which the problem occurs. This requires SLO compliance auditing across multiple provider domains. Thus, each provider involved should offer a metering service, and in case of network performance parameters, such as network delay, jitter, and packet loss, MeSA [5] (Measured Signaling for Auditing) capable routers at domain borders need to be deployed.

MeSA is a protocol which supports the measurement of delay and jitter in a multi-domain environment. It is able to map delays to domains in which they occur, but still keeping the autonomy of each single operator. Therefore MeSA capable routers append timing information to special MeSA probes which are generated periodically at the source End System. At the destination this probes are evaluated in order to detect SLA violations.

Within the VO, the access network provider is responsible for authorizing usage of the metering service. Thus, it operates an Authorization Authority (AA) component. Based on the result of the authentication and authorization process, the Network Manager (NM) component will be contacted to configure Access Routers (AR) accordingly. For the purpose of auditing of end-to-end performance, probes must be generated at one end and collected at the other end. Thus, an ES in an end-customer network comprises a MeSA Probe Generator (MPG), a MeSA Probe Collector (MPC), and an SLO Compliance Monitor (SLO CM). A more detailed view of those architectural components and their related interfaces is depicted in Figure 3..

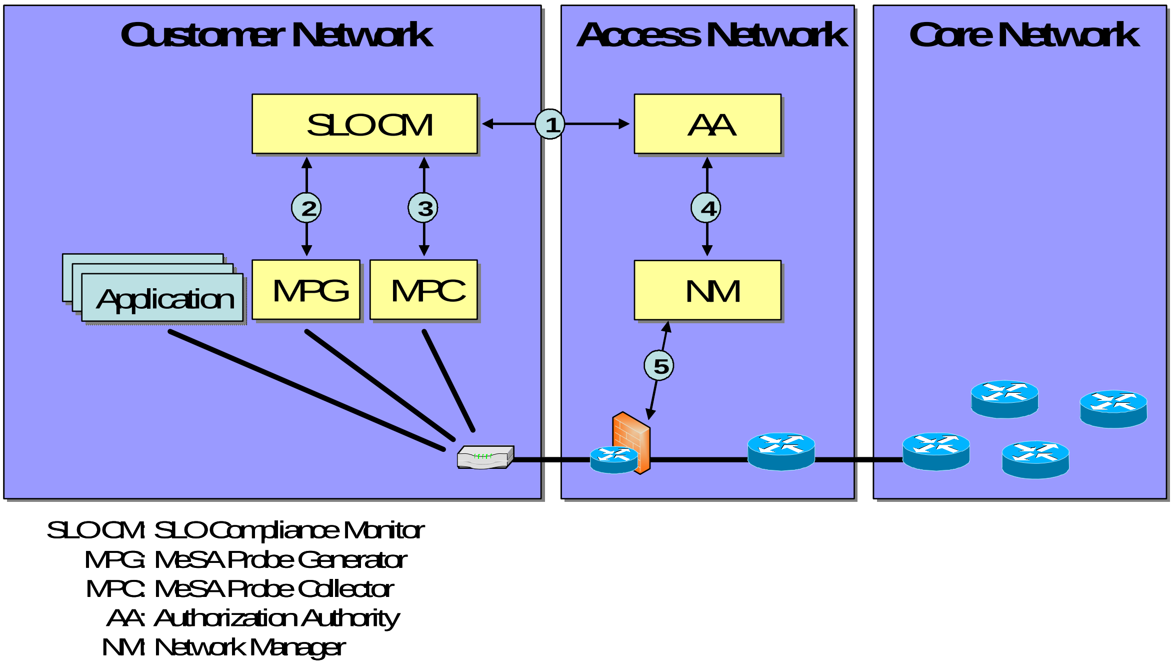

Customer Network
Access Network
Core Network
SLOCM
AA
1
2
3
4
5
Application
MPG
MPC
NM
SLOCM: SLO Compliance Monitor
MPG: MeSA Probe Generator
MPC: MeSA Probe Collector
AA: Authorization Authority
NM: Network Manager

Figure 3: ASAM Components and Interfaces

In order to describe interactions of those components in a clear way, assume an end-customer, who runs a VoIP application, wants to monitor one-way end-to-end delay in both communication directions. Assume also that the application uses port number p and service class sc_1 (which corresponds to some value of delay guarantee) for the data transport. The whole auditing process is divided into 3 phases:

1 Authorization phase
2 SLO monitoring phase
3 Tear-down phase

In the authorization phase, SLO CM requests a metering service from AA. In the request message, the service class under examination is specified, as well as the address of the communicating end systems, the customer ID, and credentials. Based on this information and the SLA concluded between this customer and the VO, AA decides on whether the request is to be accepted or rejected, and sends a reply to SLO CM accordingly. In case of a positive decision, AA requests NM to configure ARs, so that MeSA probes of a specific service class and with specific addresses are allowed to pass through and to be processed. After receiving a positive answer from AA, SLO CM configures MPG to generate MeSA probes for traffic with specific source IP address, source port number p, and service class sc_1.

In the SLO monitoring phase, MPG intersperses MeSA probes in the data packet stream sent by the VoIP application based on information received from SLO CM and the IP header. The probes are then sent to MPC at the other end system. An MPC waits for incoming probes on a pre-determined port number. Probes collected by MPC are delivered to SLO CM for further processing, namely auditing. The SLO compliance auditing process can be realized using a generic auditing framework, *e.g.*, AURIC, as presented in [10].

In the Tear-down phase the results of the SLO compliance auditing are sent to the initiating SLO CM. The initiating SLO CM may then claim SLA violations at its Access Provider.

The above mentioned interaction allows for auditing of one-way delay in direction leaving the initiating SLO CM. To obtain one-way delay of the other direction, probes have to be generated by the other end and sent to the initiating SLO CM. In order to enable this to happen, the initiating SLO CM requests MPG to send a special MeSA packet to the other end. Having received this special MeSA packet, the responding SLO CM configures its MPG to generate the requested probes.

3.3 Status

The draft architecture as outlined above has been designed, which comprises key elements of ASAM. Furthermore, initial protocol interactions for SLO compliance auditing have been outlined. Based on these achievements, the design refinement including a detailed message structure, the implementation architecture, and a relevant simulation followed by the evaluation constitute the next tasks to be accomplished in ASAM.

4 SaPDoGS: SLA and Promise Descriptions of GRID Services

SaPDoGS proposes to construct a simple model of Grid services by documenting necessary and sufficient promises to enable the collaboration between the Grid nodes. Given a promise graph for a typical Grid system, the economic behavior of the Grid can be examined by looking at exchanges and the value of each promise to each node. In this way it is foreseen whether Grids can be sustained in a peer-to-peer manner or whether they require central management.

The Scope of this activity contributes to the objectives of WP8, mainly addressing the Task 8.5:

- Investigations on SLA planning and negotiation.
- Application of promise theory onto SLAs and their economic value.

SaPDoGS intends to relate the work to the contract work presented and referred at the AIMS conference in 2007.

4.1 Introduction

This project expects to achieve an understanding of end-to-end services within a distributed environment by using a model based on promise theory. In order to understand the basic economics of a service network it is necessary to understand underlying requirements and risks in a distributed system. In a service model, all agents in a network can be considered independent and autonomous. Promise theory allows for the modeling of the effect (if any) of combining promises to see whether they are transferable from end-to-end. The Grid provides a useful backdrop for studying these issues and an example of Grid computing will be used to illustrate the work.

4.2 System Design

The system of study will be based on a general tree of nodes each of which can make promises. Necessary and sufficient conditions will be looked for to infer end-to-end promises. After this distributed risk associated with the breakage of certain promises is considered.

If time permits and necessary permissions are obtained the integration of measurements from an actual Grid system are foreseen to provide a preliminary indication as to the practicality of the economic model.

4.3 Status/Preliminary Achievements

All status known so far is that a formal model of the promises for distributed services requires a set of much stronger definitions than is usual in discussing service architectures. Precision is the key when a promise has been made: by whom and about what. It is not sufficient to assume that all of the parties involved in an end-to-end architecture will keep their promises, or be influences by the desire of the end points to have the proposed service.

5 BP3EM: Best Practices, Processes and Promises in Economic Management

In IT Management, more and more attention is paid to Best Practice recommendations and process-oriented approaches, like the IT Infrastructure Library (ITIL), as already described in detail Deliverable D8.2. This turn from a pure technological view point into IT Service Management (ITSM) from a perspective covering organizational aspects comes with various challenges.

The overall goal of the BP3EM approach is to contribute in bridging the gap between organizational ITSM approaches and the “hands-on-the-keyboard” system management and administration tasks, with a specific focus on the economic management disciplines including:

- Service Level Management
- Performance Management

Thus, the very concrete questions to ask and to address include the following key excerpt, which may be expanded during the further project progress:

- Mechanisms View Point:
  - ♦ How can processes be “translated” into management policies?
  - ♦ How can processes be mapped onto promises?
  - ♦ How can IT Service performance parameters be monitored by aggregating resource parameters?
- Application View Point:
  - ♦ What are the processes related to Performance Management?
  - ♦ What are the processes related to Service Level Management?
  - ♦ How can the automation of processes be supported?

5.1 Introduction

The emerging trend toward business-driven IT Management (BDIM) is addressed by various IT Management frameworks. One of the most popular and currently leading approaches in this area is given by the recommendations of the IT Infrastructure Library (ITIL), a public domain best practice framework for IT Service Management, owned and released by the British Office of Government Commerce (OGC) [17], [18], [19], [20], [21]. ITIL defines processes for the design, transition and operational support of IT services of almost any kind. In its newest version (ITILv3), published in 2007, the paradigm of business alignment has become an even more essential part of the ITIL guidance, which is in particular expressed by the two core titles Service Strategy [20] and Continual Service Improvement [17].

While business-driven IT Management approaches find favour with many IT providers and practitioners, this trend does not come without any deficiencies. One of the most crucial challenges in IT Service Management (ITSM) is to not lose sight of the "hands-on-the-keyboard" systems management. In other words: There is a gap between business-driven IT Service Management and traditional IT systems administration tasks from an IT Management perspective, although IT services require well-maintained resources in order to operate in an effective and efficient way. This coherency and the resulting gap are depicted in Figure 4 which exemplary illustrates the scope of ITIL as an ITSM framework and cfengine as a powerful solution framework for a variety of common network and systems administration tasks. Realizing the interfaces between ITSM and systems administration is not sufficiently addressed by current research or practical approaches and therefore one of the goals of BP3EM.

In other words:

- The scope of ITIL is much broader than traditional systems administration, *but* portions of systems administration and configuration management tasks take place in the context of certain ITIL processes.

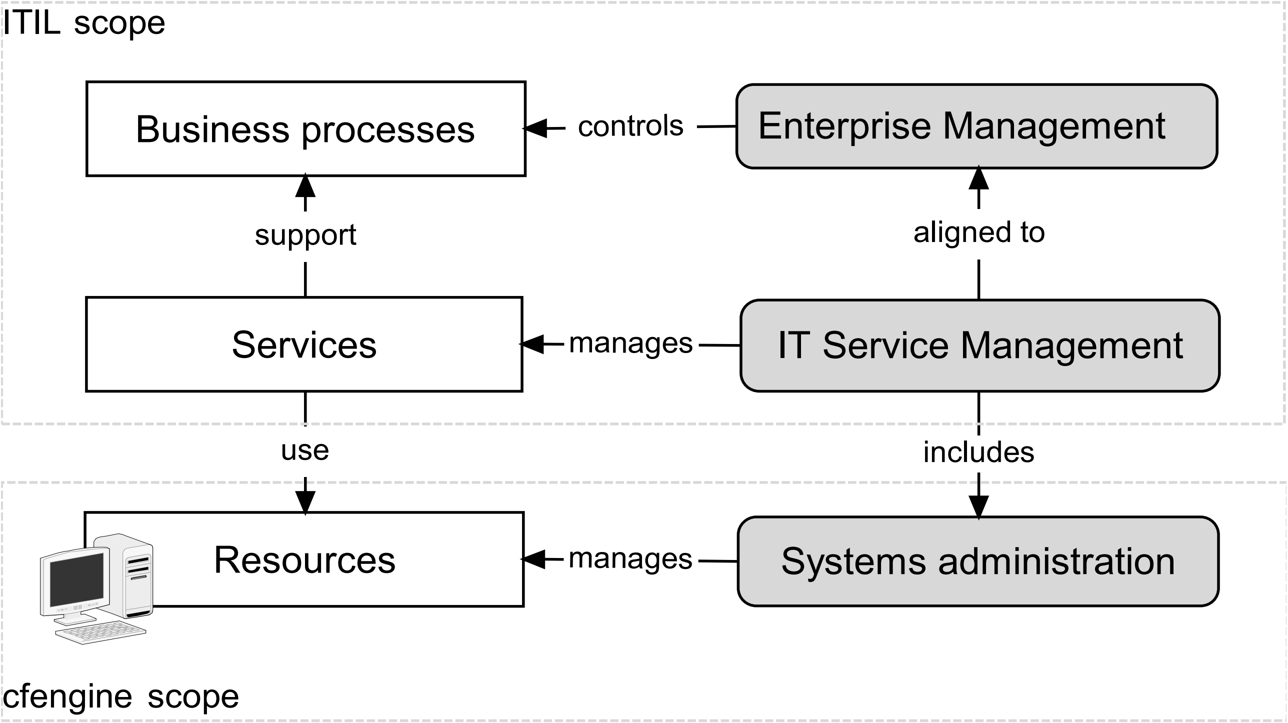


Figure 4: Gap between IT Service Management and Systems Administration

- cfengine has not been designed to replace ITSM tools like trouble ticket systems (TTS), workflow management or Configuration Management Databases (CMDBs), *but* in the more technical areas of IT Service Management, cfengine is able to support ITIL processes in their activities.

The goal of the BP3EM complementary document "Integrating cfengine with ITIL processes" (working title) that is currently under joint development of the BP3EM partner institutions, is to give an overview on how cfengine can be used to support selected IT Service Management tasks according to ITIL as en exemplary approach of bridging the gap between IT Service Management (processes) and policy-/promise-based systems administration.

5.2 System Design

In order to complete the considerations mentioned above, BP3EM collects, analyzes, and consolidates other latest research results in the context of IT Management processes, polices, and promises with a specific focus on Economic Management issues, in particular Service Level Management and Performance Management. This section gives a brief preliminary and introductory overview of two of those systems.

5.2.1 Performance Management: Service Monitoring Architecture

IT Service Monitoring is one of the challenges when bridging the gap between ITSM and resource management. A proposal for a Service Monitoring Architecture (SMONA) has been developed in [4] and is depicted in Figure 5. The layered design of this architecture aims at collecting infrastructure monitoring data through heterogeneous platforms, converting these platform-/vendor-specific data into a platform-independent format and aggregating these data to service monitoring information. A Service Information

Specification Language (SISL) is used by the Service Management/Monitoring Application. A detailed description of SMONA and the related workflows is available in [22].

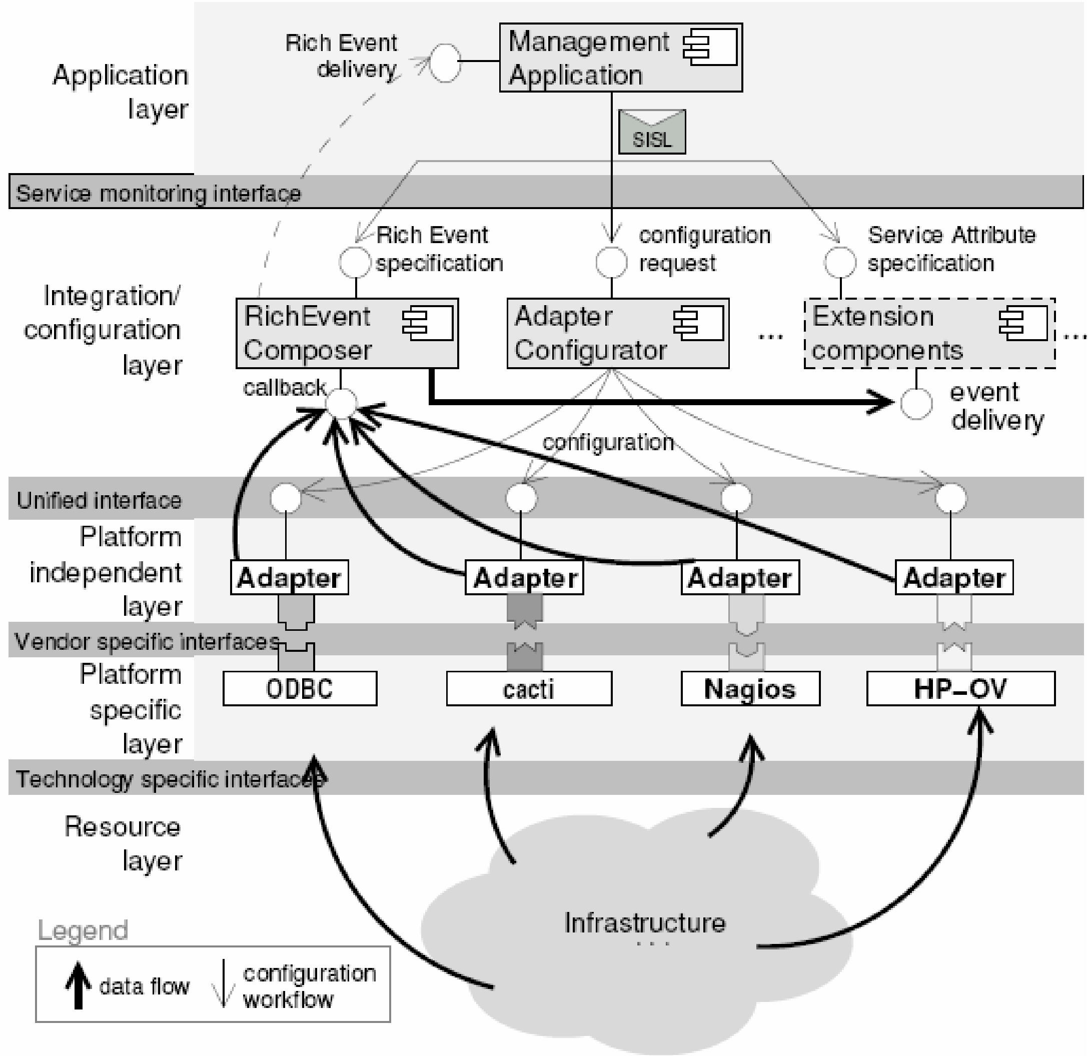


Figure 5: Service Monitoring Architecture (SMONA)

5.2.2 ITSM Modeling Framework

In order to develop tool concepts and solutions for IT Service Management, model-based approaches are gaining in popularity. Recent research results have been published in [1]. Figure 6 shows the ITSM Modeling Framework proposed in this work. Its goal is to provide guidance in requirements engineering, design and implementation of management systems/tools for IT Service Management tasks and processes.

It differentiates between three different views on the managed objects (horizontal layers) and four domain classes (vertical layers). On the Owner View, business models are provided in order to model and formalize business requirements — *e.g.*, by determining management process flows, use cases or organizational models. The Designer View provides a concrete, but yet platform-independent system (architecture) model which is refined into a platform-specific technology model on the Builder View.

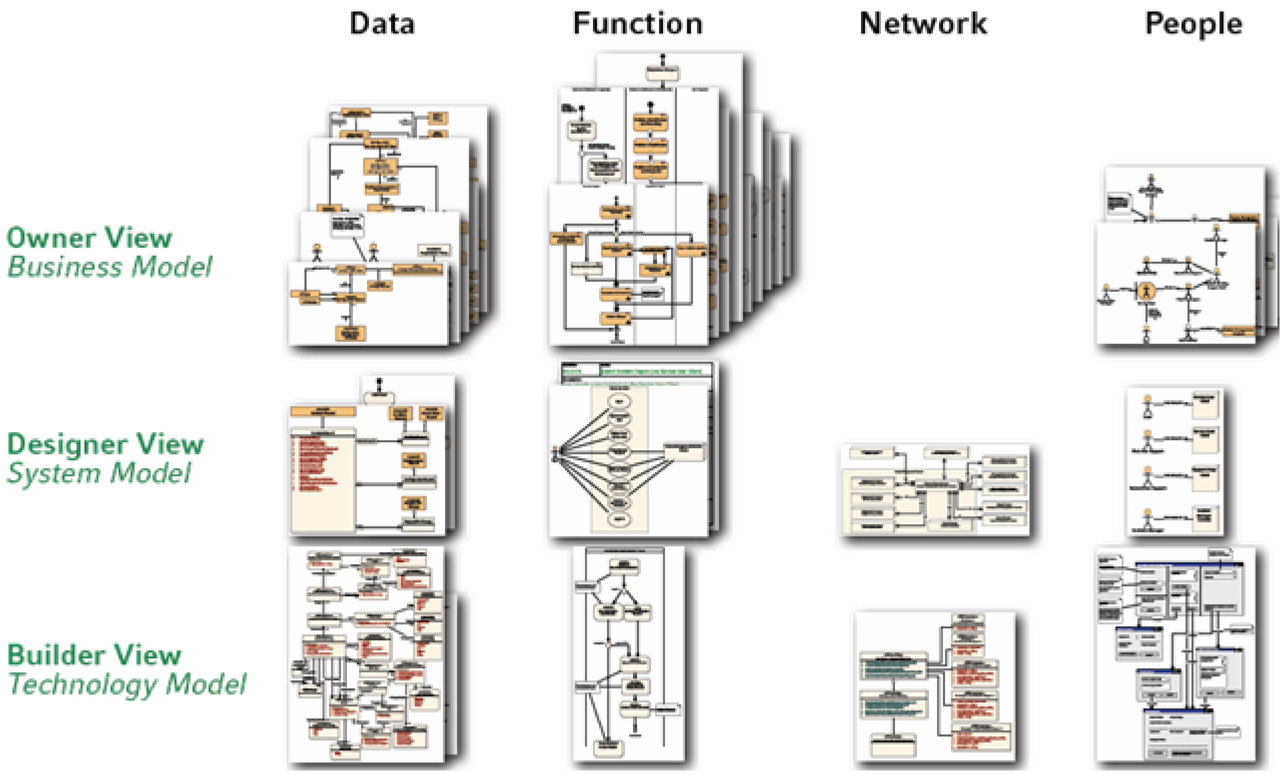


Figure 6: ITSM Modeling Framework

5.3 Status/Preliminary Achievements

Currently, foundational requirements on the expected solutions have been discussed during a face-to-face meeting of the BP3EM partners. A brief overview on the project intentions has already been given in the previous subsections. One of the key deliverables of BP3EM will be the handbook on integrating cfengine with ITIL processes as an example on how to "merge" business-driven IT Management and traditional systems administration.

6 PRIPOL: Pricing by Policies

Policy analysis and refinement have developed tools helping to design enforceable policies from high level goals as well as to detect and solve conflicts between policies.

The aim to investigate the applicability of such policies techniques to establish prices of services within a market determines the key underlying approach here. Results obtained so far are outlined below and, therefore, the continuation of this approach will include pricing and accounting expertise in a combined manner.

The idea is basically to describe the problem of getting a market price as a problem that can be formulated in terms of policies that either have to be deduced automatically from high level service provider objectives or/and get in conflict between the interest of the involved parties (the service provider with the users or with other service providers). The major area of application will consider Internet-based services only at this stage, which range from pure IP access services to value-added third party services. Thus, the focus on economic mechanisms for an extended IP network management model is being met.

Therefore, the key objective of this approach is to develop an approach, in which the derivation of prices for services makes use of policy analysis techniques.

6.1 Price, Quality, and Congestion Solving

In networks supporting different classes of services, it is expected that users will enjoy different levels of service for the traffic injected into the network. In these scenarios users should be allowed to choose between different prices aligned to the service level selected and enjoyed. Service providers need to adjust both, prices and service levels accordingly.

On the one hand, users should be encouraged by service providers to utilize network resources to exploit the managed network resources. On the other hand, the allocation of resources to assess the arranged quality of service would need to be adjusted accordingly. There is the strong believe that all of theses above adjustments should obey to a well-defined business perspective in which pricing is part of it.

Generally speaking [2], users' demand will decrease with the increase of price, while supply will increase and vice-versa. There is an equilibrium point between the demand-price and supply-price curves, where the demand and supply will be equal under that price. This phenomenon is graphically illustrated in Figure 7.

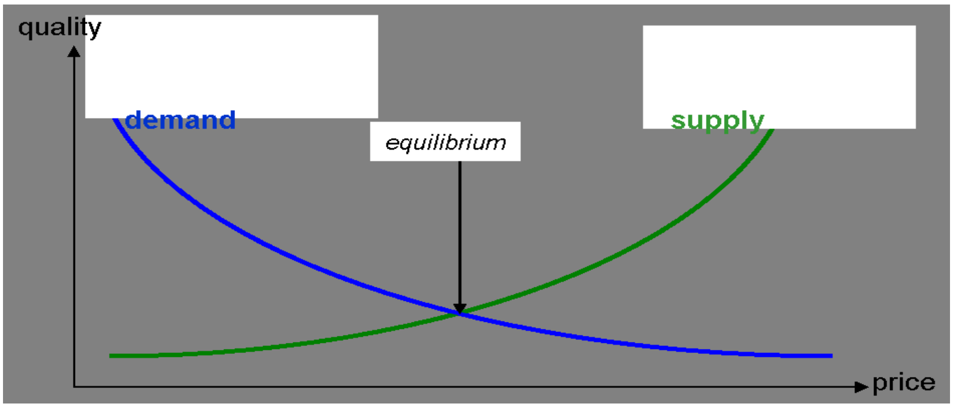


Figure 7: Price and Quality Equilibrium

Without a control of any kind (like price, admission control, or resources limitation) traffic injection would produce resources starvation and consequently congestion (see upper-left part of Figure 8). Congestion solving would uniquely be a function of the user's utilization, most probably due to customers avoiding to inject or reduce traffic injection due to service degradation. This is the common behavior, when rate and admission control mechanisms are fixed or not modified dynamically to address congestion and/or service degradation.

From a business-oriented management approach, congestion and service degradation would have impact on business. Profit, reward, and refund may be addressed due to SLA violation. In this basic scenario corrective actions are necessary. Nevertheless, corrective actions, like traffic rate adjustments, service invocation control, or network re-dimensioning, may be oriented to reduce the negative impact on that business in question .

6.2 QoS DiffServ Management Approach

In order to support the Quality-of-Service (QoS) guarantees for forthcoming services of the future, Next Generation IP Networks will make use of technologies, such as Differentiated Services (DiffServ) and Multi-Protocol Label Switching (MPLS), for traffic engineering and

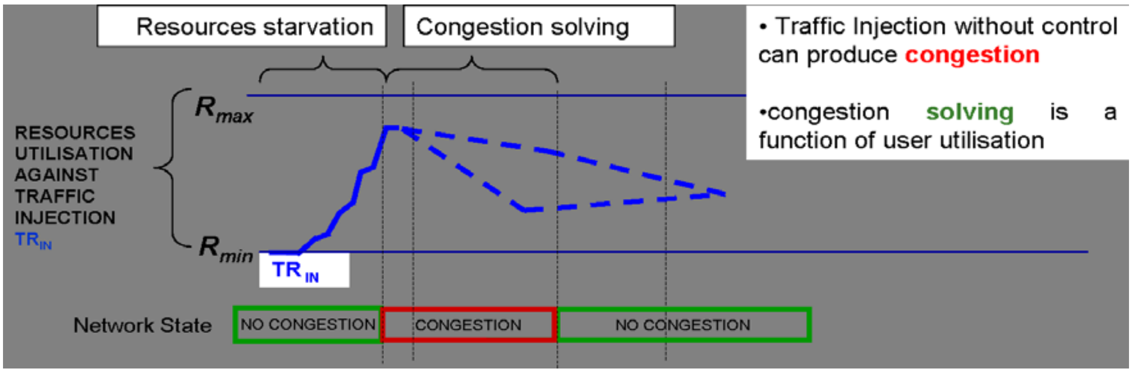


Figure 8: Resources Utilization, Traffic Injection and Network State

network-wide resource management. In addition, the volume and type of the traffic injected by these services will need to be controlled since the means to both prevent QoS degradation of active services, and verify that clients inject traffic in accordance with pre-agreed Service Level Agreements (SLAs).

In this context, QoS delivery passes through the integration of Service Management and Traffic Engineering functions as proposed in the IST TEQUILA project Traffic Engineering for Quality-of-Service for the Internet at Large Scale [23]. To the best of the authors' knowledge, this is the only approach that brings together this functionality to provide an overall architecture for QoS support in IP Networks. A simplified representation of the TEQUILA approach is depicted in Figure 9.

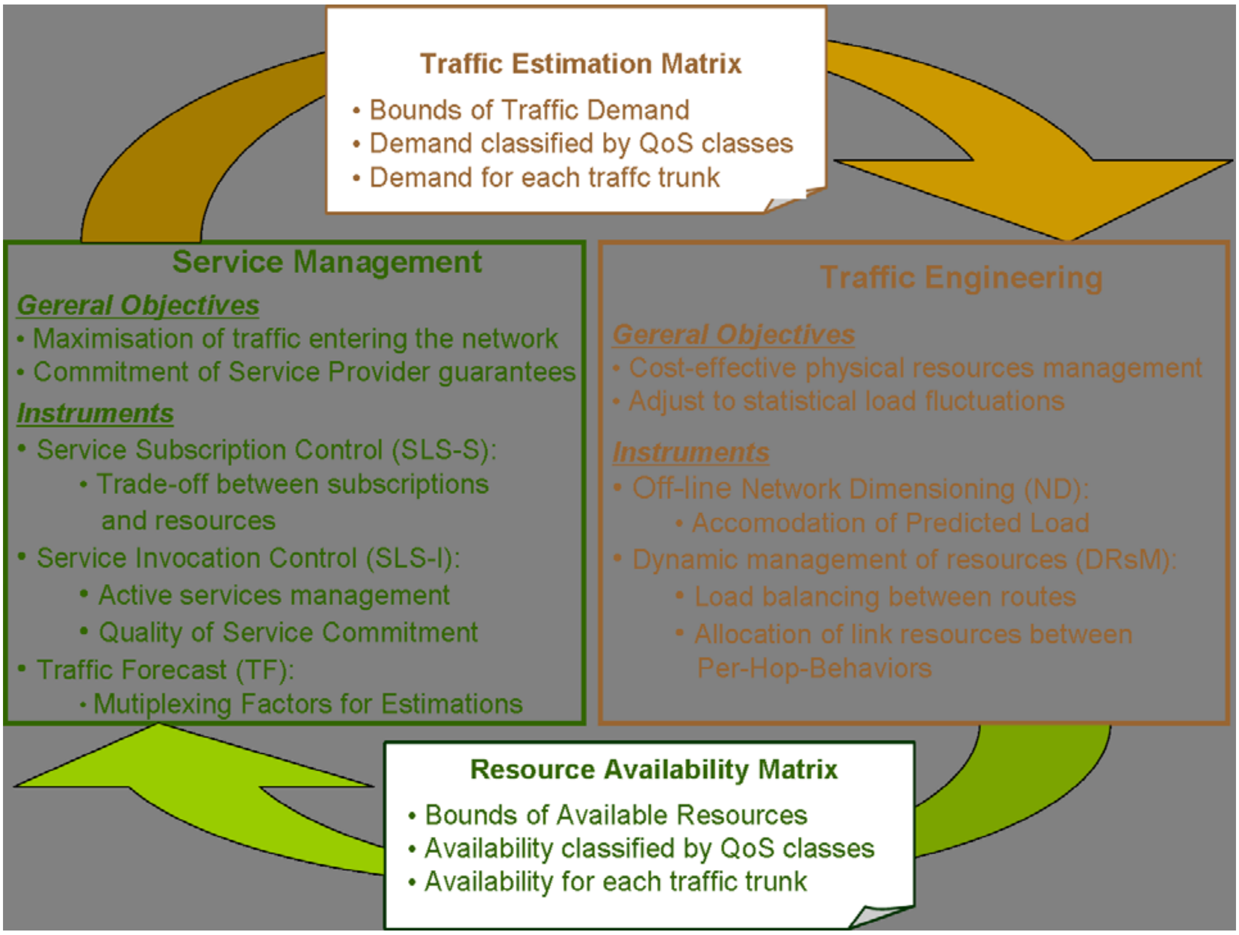

Traffic Estimation Matrix
• Bounds of Traffic Demand
• Demand classified by QoS classes
• Demand for each traffc trunk
Service Management
Geral Objectives
• Maximisation of traffic entering the network
• Commitment of Service Provider guarantees
Instruments
• Service Subscription Control (SLS-S):
• Trade-off between subscriptions and resources
• Service Invocation Control (SLS-I):
• Active services management
• Quality of Service Commitment
• Traffic Forecast (TF):
• Mutiplexing Factors for Estimations
Traffic Engineering
Geral Objectives
• Cost-effective physical resources management
• Adjust to statistical load fluctuations
Instruments
• Off-line Network Dimensioning (ND):
• Accomodation of Predicted Load
• Dynamic management of resources (DRsM):
• Load balancing between routes
• Allocation of link resources between Per-Hop-Behaviors
Resource Availability Matrix
• Bounds of Available Resources
• Availability classified by QoS classes
• Availability for each traffic trunk

Figure 9: Resources Utilization, Traffic Injection and Network State

The Service Management part has two objectives: the maximization of traffic entering the network, and the commitment of the service providers QoS guarantees. As the traffic entering the network is a function of the number of subscribed contracts and active services, admission control mechanisms are defined for service subscriptions and invocation requests. QoS commitment is addressed by enforcing preventive and corrective actions as a means to police misbehaving users, and also to resolve potential cases of network congestion.

The Traffic Engineering functionality is concerned with the management of physical network resources. An off-line dimensioning process is responsible for mapping the predicted traffic demand to the physical network resources. In addition, real-time operations are implemented as the means to first, balance the load amongst the established Label-Switched Paths (LSPs) in the network, and second, to ensure that link capacities are appropriately distributed among the different Per-Hop-Behaviors (PHBs) sharing each link. These real-time operations react dynamically to statistical traffic fluctuations.

6.3 System Design

In principle, the solution that we have initially proposed consists on a bidirectional approach between the business domain and the technological domain. While the business domain would deal with pure economical issues ranging from business objectives to pricing, the technological domain would deal with network management issues supporting the business area. A key issue in this approach is the business perspective definition. A business perspective would be used to define the high-level goals that are to be refined into polices and most probably, it would be used to define the business-oriented policy constraints for the policy refinements. At the end, the refined policies would control the network behavior aligned to the above business perspective.

We plan to use business indicators as the means to prioritize active changes that may be taken in order to prevent negative effects on the business due to statistical changes of network behavior. The business-aware policy refinement process may target the generation of policies aimed at minimizing the negative effect of the above incidents on the business indicators

6.4 Status/Preliminary Achievements

We are now working in the definition of business indicators and modeling the impact of network incidents in such business indicators. The refinement tools and mechanism are already available. Initial models are intended to be tested by simulations.

7 GridAcc: Grid Accounting

With the trend of adopting Grid systems as a mean for service-oriented computing in dynamic VOs (Virtual Organizations), the need for appropriate support mechanisms becomes apparent. Accounting of Grid resource and service usage determines the central activity as it prepares accounting records to provide the main input for analysis, optimization, and in particular for charging and billing purposes.

An embracing study of existing Grid accounting systems has revealed that these approaches focus primarily on technical precision and on project-specific issues. However, existing systems do not support multi-provider scenarios or virtualization concepts—both key requirements for service provisioning and the according accounting in dynamic Virtual Organizations. Moreover, existing Grid accounting approaches are not based on the appropriate economic accounting principles. Consequently, a resource-based, highly flexible accounting model for dynamic Virtual Organizations was developed, combining both, technical and economic accounting by means of Activity-based Costing (ABC) service constituent parts and defined accountable units.

Driven by a successful preliminary functional evaluation, the developed accounting model will undergo a full-fledged evaluation in order to draw conclusions on potential refinements and adaptations required. This is done for specific cases, such as for the environment at the Leibniz Supercomputing Centre (LRZ) [13] in Garching, Germany, which is lead by Prof. Dr. Heinz-Gerd Hegering being a member of LMU (Ludwig-Maximilians-University of Munich, Germany). For that purpose, the detailed model application and evaluation methodologies as well as considered evaluation environments need to be outlined and the actual model-based cost calculations for the set of considered Grid services defined need to be accomplished. Results gained are analyzed and respective conclusions on the model applicability and its optimizations are drawn.

7.1 Introduction

Grid service accounting constitutes the central functional support activity facilitating the creation of service and resource usage records, both, in research-oriented and in business Grid systems. Accounting relies on successful user authentication and authorization. Once that access to a resource and service, respectively is granted, resource usage has to be accounted reliably, whereupon accounting data becomes retrievable for auditing purposes or—in a fully competitive environment—it is finally transferred into charging records which in turn will be equipped by monetary values so that a bill to the service consumer can be issued. These steps are reflected by AAA (Authentication, Authorization, Accounting) [11], [14] and its extended view, A4C (Authentication, Authorization, Accounting, Auditing, and Charging) [12], [3].

Accounting for Grid systems determines an important research focus as it constitutes on the one hand the key mechanism for commercial electronic services to be offered and charged to customers. On the other hand, accounted data potentially contains valuable information for a Grid service provider about current and past service usage as well as resource consumption, what can be used for charging purposes as well as for internal optimization processes. Both uses demand for accountable units that equip a service provider with significant information that correlate with chosen optimization criteria, such as focusing on cost drivers.

Consequently, a resource-driven and activity-based accounting model for Dynamic Virtual Organizations (DVO)—as implemented by Grid systems—was developed [8], [9]. The model is used to calculate costs incurred for a given provided Grid service. There are many accounting approaches for Grid systems available, but those systems lack a sound economic basis while being highly specific so that they are not generically applicable [8]. The developed model has proven to be a highly promising approach from a functional point of view.

Based on the existing conceptual evaluation of the presented approach, a full-fledged assessment of this model in existing Grids environments needs to be undertaken. This is facilitated by applying the model to the Grid infrastructure operated and run by the Leibniz supercomputing center (LRZ) in Garching, Germany [13]. Thus, the evaluation’s main goal consists in applying the conceptually evaluated Grid accounting model to existing operational Grid infrastructure in order to reveal the key set of practical aspects relevant for model application and to determine model improvements.

7.2 System Design

Applying an extensive and flexible accounting model to a complex environment requires an elaborate methodology to be in place and followed. Figure 10 provides an overview on the respective chosen model application methodology. It is structured into two main, timely separated building blocks, ABC taking input values from TCAS (0) and IT product cost calculation (1). IT product cost calculation relies on those activity costs determined by ABC.

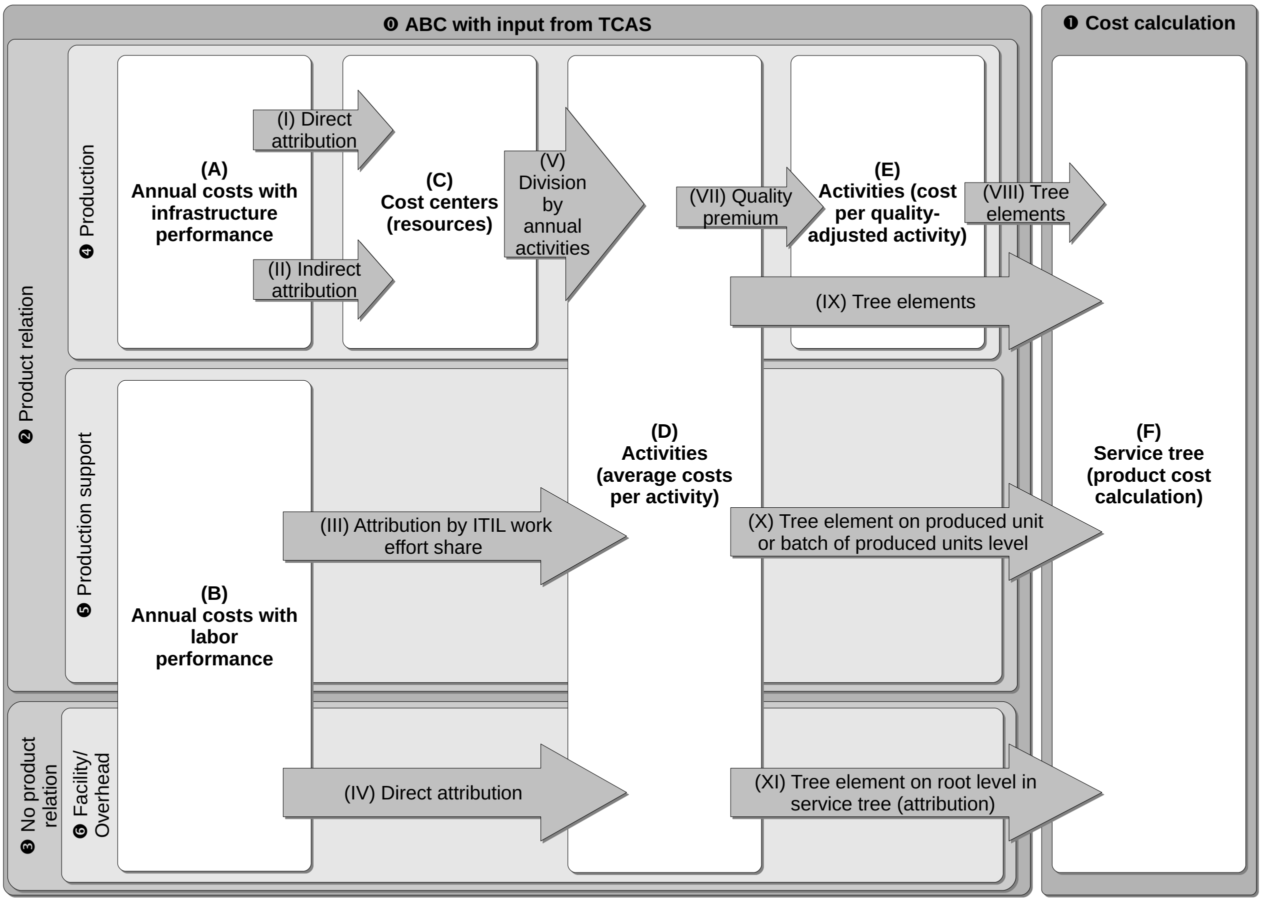

⓿ ABC with input from TCAS
❶ Cost calculation
❷ Product relation
❸ No product relation
❹ Production
❺ Production support
❻ Facility/ Overhead
(A) Annual costs with infrastructure performance
(B) Annual costs with labor performance
(C) Cost centers (resources)
(D) Activities (average costs per activity)
(E) Activities (cost per quality-adjusted activity)
(F) Service tree (product cost calculation)
(I) Direct attribution
(II) Indirect attribution
(III) Attribution by ITIL work effort share
(IV) Direct attribution
(V) Division by annual activities
(VII) Quality premium
(VIII) Tree elements
(IX) Tree elements
(X) Tree element on produced unit or batch of produced units level
(XI) Tree element on root level in service tree (attribution)

Figure 10: Grid Accounting Model Overview

ABC seeks to identify costs per activity. In the applied methodology, activities are grouped after the criterion whether they can be related to an IT product (2) or they lack a product relation (3). Activities with product relation are further grouped in production activities (4) and activities that support production (5). The first covers activities as determined by resource-specific instantiations of those introduced service constituent parts, namely *Processing*, *Storage*, *Transferring*, *Output*, *External*, and *Other*. The latter includes activities such as IT service and infrastructure management. Activities without product relation typically embrace facility management and administrative tasks (6).

The accounting model takes annual costs of various types as input. These cost elements constitute typical values of a Traditional Cost Accounting System (TCAS). In the area of production-oriented activities, input values are needed in terms of annual costs with infrastructure performance (A). This is due to the fact that IT production in this context means the provisioning and composition of electronic services, such as a storage service. These services, out of which the final IT product is composed, are provided on infrastructure, that is, on IT resources.

A given annual cost element with infrastructure performance is either attributed directly to the specific resource it relates to (I) or—in case these costs are not attributable to one of the existing IT resources—that cost element needs to be attributed indirectly by means of an allocation base, which is bound to an additional cost-relevant characteristic (II). IT resources, thus, reflect a concept from TCAS, namely the idea of a cost center. These cost centers embrace LRZ-internal computing and storage resources (C).

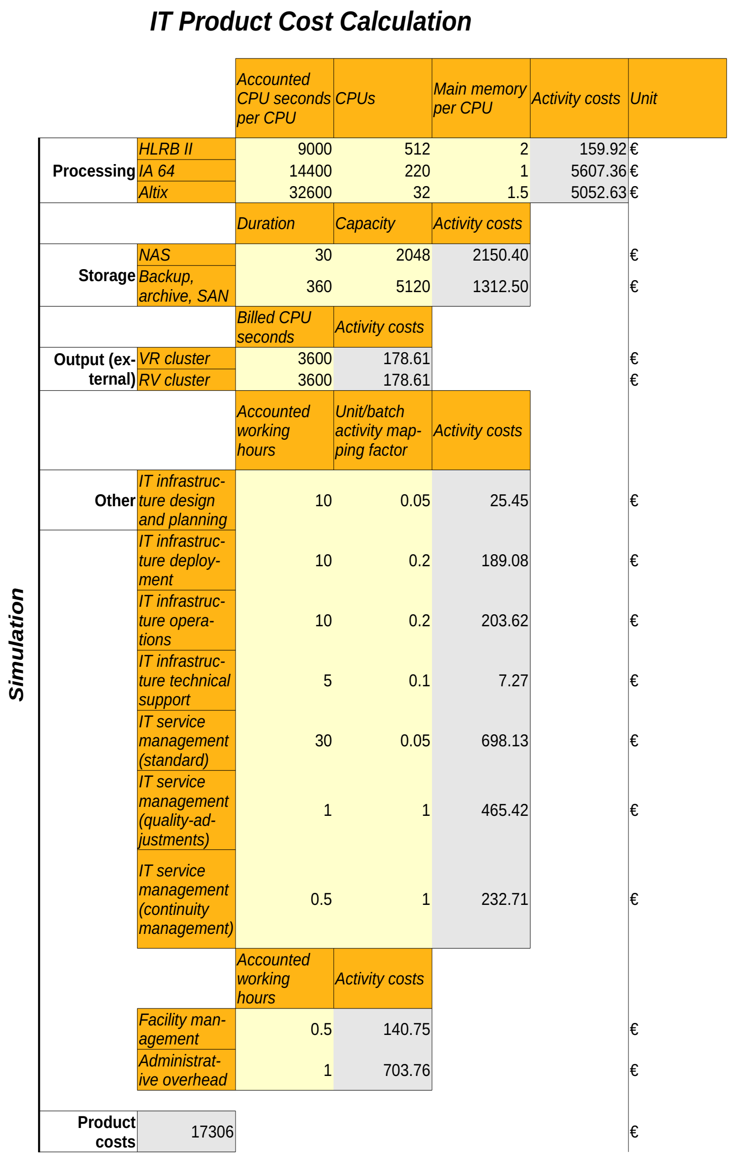
After direct (I) or indirect (II) attribution of annual costs with infrastructure performance (A), total annual costs per considered IT resource—each representing a cost center—are revealed (C). Total annual costs per resource are defined as the sum of all direct annual cost elements and all indirect annual cost elements. The latter is attributed according to the respective annual cost share for air conditioning, emergency system, network infrastructure, and building costs.

In contrast to annual costs with infrastructure performance (A), annual costs with labor performance (B) do not require an intermediate attribution step to cost centers, i.e. resources, since labor performance costs are directly related to activities (D). Annual costs with labor performance (B) and production support (5) cover human labor activities grouped after the IT Infrastructure Library (ITIL) [16] version 2. These best practices determine the de-facto standard in service management. The respective books on infrastructure and service managements are of particular importance for this work as they are concerned with production support activities.

Relative shares are used as keys to attribute (III) annual costs with labor performance (B) and production support (5) to the respective ABC activities (D). Annual costs with labor performance (B) without product relation (3) include facility management and administrative overhead activities. For both types, average costs per activity (D) are directly retrievable (IV), i.e., an attribution according to a key is not necessary.

15 activities (D) result from either dividing resource-attributed costs by annual activities (V) or attributing annual costs with labor performance (B) by either ITIL work effort share (III) or by direct attribution (IV). Each activity is bound to a service constituent part. Production activities (4) are represented by a *Processing*, *Storage*, or external *Output* service constituent part, while production support (5) and facility/overhead activities (6) are represented by the *Other* service constituent part. This list of activities constitutes the key functional step in applying the Grid accounting model as it comprises those activities that form the basis for ABC. At this step in model application (D), firstly the full list of activities to build a service tree (F) from is available, and secondly average costs per activity are revealed.

These activities can either directly (VIII to XI) form elements of the service tree (F) for product cost calculation (1) or, before that, they can be further refined in order to support quality adjustments (E). Quality-adjusted activities are determined for all internal activities, thus, for all Processing and Storage activities. The underlying principle for quality adjustments funds on a quality premium scheme. It supposes that non-adjusted activities (D) include a standard configuration.



IT Product Cost Calculation

			Accounted CPU seconds per CPU	CPUs	Main memory per CPU	Activity costs	Unit
Simulation	Processing	HLRB II	9000	512	2	159.92	€
		IA 64	14400	220	1	5607.36	€
		Altix	32600	32	1.5	5052.63	€
			Duration	Capacity	Activity costs		
	Storage	NAS	30	2048	2150.40		€
		Backup, archive, SAN	360	5120	1312.50		€
			Billed CPU seconds	Activity costs			
	Output (external)	VR cluster	3600	178.61			€
		RV cluster	3600	178.61			€
			Accounted working hours	Unit/batch activity mapping factor	Activity costs		
	Other	IT infrastructure design and planning	10	0.05	25.45		€
		IT infrastructure deployment	10	0.2	189.08		€
		IT infrastructure operations	10	0.2	203.62		€
		IT infrastructure technical support	5	0.1	7.27		€
		IT service management (standard)	30	0.05	698.13		€
		IT service management (quality-adjustments)	1	1	465.42		€
		IT service management (continuity management)	0.5	1	232.71		€
			Accounted working hours	Activity costs			
		Facility management	0.5	140.75			€
		Administrative overhead	1	703.76			€
	Product costs	17306					€

Figure 11: Grid Accounting Calculation Excerpt

7.3 Preliminary Achievements

Figure 11 documents these steps for an IT product cost calculation—here in terms of a Grid service. These steps are based on the respective outcome of those various measures taken as described in Section 7.2.

Thus, after having identified and collected necessary annual cost elements, both with either infrastructure or labor performance, and after having determined and characterized production as well as production support activities, product costs can be calculated. Figure 11 shows this for a special multi-provider Grid scenario developed, resulting in the according information that each provisioning of that scenario's core service needs to cover costs of 17.306 €.

8 Summary and Preliminary Conclusions

The current state of work within WP8 shows that those 5 projects are at different states of work. While 4 out of 5 have achieved a set of intermediate results (as documented above in the respective sections and summarized below) one project is at its beginning.

Through the establishment of Virtual Organizations (VOs) from several co-operating service providers (including network providers and value added service providers) the capability to offer end-to-end service performance becomes viable. In this case, however, monitoring of end-to-end service performance becomes a necessity as well. Based on the MeSA (Measured Signalling for Auditing) protocol and a generic auditing framework termed AURIC [10], ASAM architecture has been designed to provide this end-to-end performance metering and auditing across multiple provider domains in an efficient and effective manner.

The objective of BP3EM is to find ways of bridging the gap between traditional Network and Systems Management and business-driven IT Service Management (ITSM). Aiming at the provision of a practical, application-oriented solution, BP3EM examines the potential of integrating a popular process-oriented ITSM framework — the IT Infrastructure Library — with cfengine as a common policy-based configuration management tool. Vital for leading this project to success is the consideration of other techniques and concepts in the area of ITSM automation and tool support.

The goal of current PRIPOL's work is to deduce and propose an appropriate pricing strategy for DiffServ-based QoS domains, taking policy-based management as a pivotal element. Policy refinements determine the mechanism to be uses, accompanied with appropriate modelling of impacts of network incidents on a set of business indicators.

Work in the area of Grid accounting demonstrates the practical feasibility of successfully integrating the respective so far separated viewpoints of technical and management accounting on Grid systems. The accordingly developed Grid accounting model is applied to LRZ's infrastructure by means of a fictional multi-provider VO scenario. Most prominently, this application reveals that the resulting full cost calculation constitutes a valuable instrument, both for Grid service cost calculations as well as for service provisioning optimizations.

Thus, the dedicated set of aspects under investigation show that an integrated IP Network Management Model can be populated with important enhancements, which go beyond the traditional models of mechanisms and functionality. Therefore, WP8 on Economic Management is on the right path of advances.

9 Glossary

This section outlines again the major terms, which form the basis of those key multi-provider, and SLM models as well as of those service provisioning concepts described in this document. Additionally, it has been extended by those terms of relevance from D8.2.

- **Agreement**
  A common understanding about knowledge that is shared between two parties. Agreements are often assumed to be about policies or actions (agreements to act) and are often formalized using contracts in which case both parties agree to the terms of a common contract.

- **Architecture**
  An Architecture describes interactions of components of a complex system. Often, Architectures provide a layered or comparably structured view on the respective system.
- **Contract**
  a bilateral bundle of promises between two agents, that is intended to serve as the body of an agreement.
- **Framework**
  A Framework represents a reusable design for a system by describing concepts and structures that give guidance for the execution of a system's complex tasks without providing strict and mandatory implementation requirements or specifications. A Framework is often less concrete than a Model and described in a natural language rather than by using formal modeling techniques.
- **Model**
  A model is a representation or description designed to illustrate the structure or method of operation of an object, system or concept. In this capability, models are often used to simplify, down-scale and/or abstract from real-world entities.
- **Multi-domain**
  Adjective designating the characteristic (*e.g.*, of a management framework), that more than one administrative domain is involved. These domains often have to establish cooperation agreements on a peer to peer basis, coordinating aspects like configuration management or interaction strategies.
- **Multi-provider**
  Adjective designating the characteristic (*e.g.,* of a management framework), that more than one provider is involved. Multi-provider scenarios usually entail technical, operational and economic or legal cooperation issues between the participants to be solved by agreements.
- **Policies**
  A policy defines a course or method of action selected among alternatives and in light of given conditions to guide and determine present and future decisions.
- **Promise Agreement**
  A promise agreement is a pair of promises between two parties to acknowledge the content of an contract body.
- **Service**
  A service is the entity or unit of work offered by a service provider on behalf of a service consumer who can use the service. In general, a service includes several types of resources, including hardware- and software resources such as computing power, network links, storage capacity, and content, and it may even be composed of several sub-services.
- **Service Catalog**
  A Service Catalogue contains definitions of standard services as well as documentations of customer-specific services. It can be used as a foundation for automated service subscription or for the negotiation of SLAs.
- **Service Level Agreement (SLA)**
  A Service Level Agreement (SLA) defines the terms under which a service is offered to a service customer at a specific Service Access Point (SAP). The SLA includes a set of parameters which specify the service and the QoS under which it is provided (*e.g.*, the amount of bandwidth allocated, the involved session partners, metrics and algorithms

that are used to compute SLA parameters), accountable units and the tariff which is used to charge for the service usage. Besides, several other aspects such as penalties or actions, respectively, to be taken if SLA objectives (*i.e.,* guarantees) are violated, trust relationships are part of a SLA.

- **Service Provisioning**
  IT services can be associated with a service life cycle that subsumes the steps from planning to termination of a particular service. A popular simple service life cycle is called Plan-Build-Run, typically rerun after a Change or Improvement step. Service Provisioning mainly deals with the plan, build and change parts, providing the necessary input for run time operations. It is therefore a major part of the service life cycle. More precisely, service provisioning includes tasks such as planning new services, building the basic infrastructure, SLA negotiation and order processing, identifying adequate resources for service delivery or adapting existing services to specific customer needs, specifying steps for service implementation and service operation, up to dynamic, near real-time service composition out of service modules based on customer requirements.
- **Virtual Organization (VO)**
  A Virtual Organization (VO) is a form of organization abstraction. It is understood as a temporary or permanent coalition of geographically dispersed individuals, groups, organizational units or entire organizations that share resources, capabilities and information to achieve common objectives. VOs can provide services and thus, can take the role of a service provider.

10 References

[1] M. Brenner: *Werkzeugunterstützung für ITIL-orientiertes Dienstmanagement — Ein modellbasierter Ansatz;* Ludwig-Maximilians-Universität München, July 2007.

[2] X. Chang, D. W. Petr: *Survey of Pricing for Integrated Service Networks*. Elsevier Computer Communications, Volume 24, Number 18, pp. 1808-1818(11), December 2001.

[3] Daidalos: *A4C Framework Design Specification*; Daidalos Deliverable Update D341, September 2004.

[4] V. Danciu, N. Gentschen Felde, M. Sailer: *Declarative specification of service management attributes*; Moving From Bits to Business Value: Proceedings of the 2007 Integrated Management Symposium, IFIP/IEEE, May 2007.

[5] F. Eyermann, B. Stiller: *A Protocol to Support Multi-domain Auditing of Internet-based Transport Services*; Second International Conference on Internet Monitoring and Protection (ICIMP 2007), July 2007.

[6] I. Foster, C. Kesselman, J. M. Nick, S. Tuecke: *Grid Services for Distributed System Integration*; IEEE Computer Magazine, Vol. 35, No. 6, pp. 37-46, June 2002.

[7] I. Foster, C. Kesselman, S. Tuecke: *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*; International Journal on High-Performance Computing Applications, Vol. 15, No. 3, 2001, pp. 200-222; http://www.globus.org/research/papers/anatomy.pdf (current June 2002).

[8] M. Göhner, M. Waldburger, F. Gubler, G. Dreo Rodosek, B. Stiller: *An Accounting Model for Dynamic Virtual Organizations*; Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007); Rio de Janeiro, Brazil, May 2007, pp. 1-8.

[9] F. Gubler: *Accountable Units for Grid Services in Mobile Dynamic Virtual Organizations*; IFI diploma thesis, University of Zurich, Zurich, Switzerland, March 2006, pp. 1-97.

[10] Hasan, Burkhard Stiller: *AURIC: A Scalable and Highly Reusable SLA Compliance Auditing Framework*; 18th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2007), Springer, pages 203-215, San Jose, USA, October 2007.

[11] The Internet Engineering Task Force (IETF): *Authentication, Authorization, Accounting (AAA) Charter*; http://www.ietf.org/html.charters/aaa-charter.html, September 2004.

[12] J. Jähnert, J. Zhou, R. L. Aguiar, V. Marques, M. Wetterwald, E. Melin, J. I. Moreno, A. Cuevas, M. Liebsch, R. Schmitz, P. Pacyna, T. Melia, P. Kurtansky, Hasan, D. Singh, S. Zander, H. J. Einsiedler, B. Stiller: *The ‘pure-IP’ Moby Dick 4G Architecture*; Computer Communications, Volume 28, No. 9, June 2005, pp. 1014-1027.

[13] Leibniz-Rechenzentrum (LRZ): *LRZ Grid Portal*; http://www.Grid.lrz.de/en/overview.html, February 2007.

[14] S. Mullen, M. Crawford, M. Lorch, D. Skow: *Site Requirements for Grid Authentication, Authorization and Accounting*; Global Grid Forum GFD-I.032, http://www.ggf.org/documents/GFD.32.txt, October 2004.

[15] J. von Neumann, O. Morgenstern: *Theory of Games and Economic Behavior*; Princeton University Press, Princeton, 1944.

[16] Office of Government Commerce (OGC): *ITIL*; http://www.ogc.gov.uk/guidance_itil.asp, June 2007.

[17] Office of Government Commerce (OGC): *ITIL Continual Service Improvement*; The Stationery Office Books, May 2007.

[18] Office of Government Commerce (OGC): *ITIL Service Design*; The Stationery Office Books, May 2007.

[19] Office of Government Commerce (OGC): *ITIL Service Operation*; The Stationery Office Books, May 2007.

[20] Office of Government Commerce (OGC): *ITIL Service Strategy*; The Stationery Office Books, May 2007.

[21] Office of Government Commerce (OGC): *ITIL Service Transition*; The Stationery Office Books, May 2007.

[22] M. Sailer: *Konzeption einer Service-MIB — Analyse und Spezifikation dienstorientierter Managementinformation*; Ludwig-Maximilians-Universität München, July 2007.

[23] P. Trimintzios, I. Andrikopoulos, G. Pavlou, P. Flegkas, D. Griffin, P. Georgatsos, D. Goderis, Y. T'Joens, L. Georgiadis, C. Jacquenet, R. Egan: *A Management and Control Architecture for Providing IP Differentiated Services in MPLS-Based Networks*; IEEE Communications Magazine, Volume 39, Number 5, May 2001.

11 Abbreviations

A4C	Authentication, Authorization, Accounting, Auditing, and Charging
AA	Authorization Authority
AAA	Authentication, Authorization, and Accounting
ABC	Activity-based Costing
AR	Access Router
ASAM	Auditing of SLOs Across Multiple Provider Domains

AURIC	Auditing Framework for Internet Services
BDIM	Business-driven IT Management
BP3EM	Best Practices, Processes and Promises in Economic Management
BR	Border Router
DiffServ	Differentiated Services
DVO	Dynamic Virtual Organization
DX.Y	Deliverable X.Y
ES	End System
GridAcc	Grid Accounting
ID	Identification/Identity/Identifier
IP	Internet Protocol
ISP	Internet Service Provider
ITSM	IT Service Management
IT	Information Technology
ITIL	IT Infrastructure Library
LRZ	Leibniz Rechenzentrum, Leibniz Supercomputing Center
LSP	Label-switched Path
MeSA	Measured Signaling for Auditing
MPC	MeSA Probe Collector
MPG	MeSA Probe Generator
MPLS	Multi-protocol Label Switching
NM	Network Manager
OGC	Office of Government Commerce
PHB	Per-hop Behavior
PRIPOL	Pricing by Policies
QoS	Quality-of-Service
SaPDoGS	SLA and Promise Descriptions of GRID Services
SISL	Service Information Specification Language
SLA	Service Level Agreement
SLM	Service Level Management
SLO	Service Level Objective
SLO CM	SLO Compliance Monitor
SMONA	Service Monitoring Architecture
TCAS	Traditional Cost Accounting System
TTS	Trouble Ticket System
VO	Virtual Organization
VoIP	Voice over IP
WP	Work Package

12 Acknowledgements

This deliverable was made possible due to the large help of the WP8 team of the EMANICS team within the NoE, which includes besides the deliverable authors as indicated in the document control, Michael Brenner, Vitalian A. Danciu, Javier Rubio, and Martin Sailer. Many thanks to all of them.

13 Selected Cooperation Work

This section of selected cooperation work covers a range of work started recently in the context of EMANICS WP8 as well as sometimes shortly before. Thus, the content in this section consists out of paper abstracts and summaries from single institutions (early starts) as well as joint work between EMANICS partners (recent starts).

To provide an overview of these areas of work, the following subsections list authors, abstracts, and the table of content, if they exist, or a respective sketch of the idea to be worked on in the next month of EMANICS. In case of a full paper being available, it is part of this deliverable at its final end, following the sequence as their summaries below.

Thus, D8.3 covers 2 single affiliation papers and 1 joint affiliation papers being ready and 3 joint affiliation papers being under preparation, totalling 6 papers.

A special note has to be named explicitly: The pager in Section 13.2 has been awarded the ***"Best Student Paper Award"*** in the respective conference (IEEE/IFIP DSOM 2007). And additionally, the other paper in Section 13.4 has been awarded the ***"Best Paper Award"*** in the respective conference (IARIA ICIMP 2007).

13.1 AURIC: A Scalable and Highly Reusable SLA Compliance Auditing Framework

Authors: Hasan, Burkhard Stiller, University of Zurich

Abstract:

Service Level Agreements (SLA) are needed to allow business interactions to rely on Internet services. Service Level Objectives (SLO) specify the committed performance level of a service. Thus, SLA compliance auditing aims at verifying these commitments. Since SLOs for various application services and end-to-end performance definitions vary largely, *automated* auditing of SLA compliances poses the challenge to an auditing framework. Moreover, end-to-end performance data are potentially large for a provider with many customers. Therefore, this paper presents a *scalable* and *highly reusable* auditing framework and a prototype, termed *AURIC* (***Au***diting F***r***amework for ***I***nternet Servi***c***es), whose components can be distributed across different domains.

Table of Contents:

Publication:

Hasan, B. Stiller: *AURIC: A Scalable and Highly Reusable SLA Compliance Auditing Framework.* Lecture Notes in Computer Science, Vol. 4785, Springer 2007, pages 203-215, 18th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2007), San Jose, California, U.S.A., October 2007.

13.2 LINUBIA: A Linux-supported User-Based IP Accounting

Authors: Cristian Morariu, Manuel Feier, Burkhard Stiller, University of Zurich

Abstract:

Obtaining information about the usage of network ressources by individual users forms the basis for establishing network billing systems or network management operations. While there are already widely used accounting techniques available for measuring IP network traffic on a per-host basis, there is no adequate solution for accounting per-user network activities on a multiuser operating system. This work provides a survey on existing approaches to this problem and identifies requirements for a user-based IP accounting module. It develops a suitable software architecture LINUBIA and proposes a prototypical implementation for the Linux 2.6 operating system, which is capable of providing per-user accounting for both the IPv4 and the IPv6 protocol.

Table of Contents:

Publication:

C. Morariu, M. Feier, B. Stiller: *LINUBIA: A Linux-supported User-Based IP Accounting.* Lecture Notes in Computer Science, Vol. 4785, Springer 2007, pages 229-241, 18th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2007), San Jose, California, U.S.A., October 2007.

Remark:

This paper has been awarded the ***"Best Student Paper Award"*** in this conference.

13.3 Evaluation of an Accounting Model for Dynamic Virtual Organizations

Authors: Martin Waldburger, University of Zürich, Matthias Göhner, University of Federal Armed Forces Munich, Helmut Reiser, Leibniz Supercomputing Centre, Gabi Dreo Rodosek, University of Federal Armed Forces Munich, Burkhard Stiller, University of Zürich.

Abstract:

With the ongoing trend of adopting Grid systems as a means for service-oriented computing in dynamic Virtual Organizations (VO), the need for appropriate support mechanisms becomes apparent. Accounting of Grid resource and service usage determines the central support activity as it prepares accounting records that provide the main input for analysis, optimization, and in particular for charging and billing purposes.

An embracing study of existing Grid accounting systems has revealed that these approaches focus primarily on technical precision and on project-specific issues. However, existing systems do not support multi-provider scenarios or virtualization concepts—both key requirements for service provisioning and the according accounting in dynamic Virtual Organizations. Moreover, existing Grid accounting approaches are not based on the appropriate economic accounting principles. Consequently, a resource-based, highly flexible accounting model for dynamic Virtual Organizations was developed, combining both, technical and economic accounting by means of Activity-based Costing (ABC) service constituent parts and defined accountable units.

Driven by a successful preliminary functional evaluation, this paper pursues a full-fledged evaluation of the developed Grid accounting model. This is done for the specific environment of the Leibniz Supercomputing Centre (LRZ) in Garching near Munich, Germany. For that purpose, the detailed evaluation methodology as well as the evaluation environment is outlined, what leads to actual model-based cost calculations for a defined set of considered Grid services. Gained results are analyzed and the respective conclusions on model applicability and potential optimizations are drawn.

Table of Contents:

Publication:

in progress

13.4 A Protocol to Support Multi-domain Auditing of Internet-based Transport Services

Authors: Frank Eyermann, Universität der Bundeswehr, München; Burkhard Stiller, University of Zürich

Abstract:

Auditing of Service Level Agreements (SLA) defines the process of monitoring whether a service provider delivers agreed upon service levels or not. While frameworks exist to monitor application-level SLAs, the end-to-end monitoring of IP-carrying SLAs, especially in a multi-domain environment like the Internet, is still an open issue. This work analyzes methodologies how IP-carrying SLA parameters could be efficiently measured and develops a protocol, which supports this process and minimizes overhead.

Table of Contents:

Publication:

2nd International Conference on Internet Monitoring and Protection (ICIMP 2007), Silicon Valley, U.S.A., July 1-6, 2007, Organized by International Academy, Research, and Industry Association (IARIA).

Remark:

This paper has been awarded the ***"Best Paper Award"*** in this conference.

13.5 Business-oriented Policy Constraints for QoS DiffServ Management

Authors: UPC, KTH, UniZH

Abstract:

In networks supporting different classes of services it is expected that users will enjoy different levels of service for the traffic injected into the network. In these scenarios users should be allowed to choose between different prices aligned to the enjoyed service level. Service providers need to adjust both, prices and service levels accordingly. On one hand, users should be encouraged by service providers to utilize network resources to exploit the managed network resources. On the other hand, the allocation of resources to assess the arranged quality of service would need to be adjusted accordingly. We strongly believe that all of the above adjustments should obey to a well-defined business perspective in which pricing is part of it.

Table of Contents:

- Introduction
  - ♦ Price, quality and congestion solving
  - ♦ QoS DiffServ Management approach
- Business-oriented Policy Constraints for QoS Management
- Related Work
- Summary and Conclusions

Publication:

in progress

13.6 Policy Refinement in Pricing Scenarios

Authors: Xiaozhi Liu, KTH (Master Thesis supervised by Rolf Stadler and Joan Serrat)

Abstract:

This work addresses the feasibility of applying policy refinement to solve the pricing problem. The mechanism follows the methodology defined by the policy refinement framework, Goremoch.

We perform both roles that are required by the policy refinement process, System Developer and System Administrator. Chapter 3 and 4 are the process of System Developer, in which we design the managed system Pricing System and the goal graph containing all possible policies that can be enforced in the managed system. Chapter 5 we play the role of System Administrator. By simulation, we prove the runtime of the policy refinement process.

Table of Contents:

- Introduction
- Involved Technology Background

- Pricing Manager Architecture Design
- Pricing System Policy Design
- Simulation
- Conclusion
- Future Work
- References

Publication:

In progress

AURIC: A Scalable and Highly Reusable SLA Compliance Auditing Framework

Hasan, Burkhard Stiller

Computer Science Department IFI, University of Zürich, Switzerland
[hasan|stiller]@ifi.uzh.ch

Abstract. Service Level Agreements (SLA) are needed to allow business interactions to rely on Internet services. Service Level Objectives (SLO) specify the committed performance level of a service. Thus, SLA compliance auditing aims at verifying these commitments. Since SLOs for various application services and end-to-end performance definitions vary largely, *automated* auditing of SLA compliances poses the challenge to an auditing framework. Moreover, end-to-end performance data are potentially large for a provider with many customers. Therefore, this paper presents a *scalable* and *highly reusable* auditing framework and a prototype, termed *AURIC* (***Au****diting* F***r****amework for* ***I****nternet Servi****c****es*), whose components can be distributed across different domains.

1 Introduction

Today, the Internet has become a platform for business. Various services are offered to enable business transactions to be accomplished. A *Service Level Agreement* (SLA) is negotiated between a provider and a customer in order to define a legally binding contract regarding the service delivery. While the TeleManagement Forum defines an SLA as “a formal negotiated agreement between two parties, sometimes called a Service Level Guarantee, it is a contract (or part of one) that exists between the service provider and the customer, designed to create a common understanding about services, priorities, responsibilities, etc.” [17], in general, an SLA comprises in particular a service description, the *expected performance level* of the service, the procedure for reporting problems, the *time-frame for response and problem resolution*, the process for monitoring and reporting the service level, the consequences for the provider not meeting its obligations, and escape clauses and constraints [18]. The performance level of a service committed is specified in a set of *Service Level Objectives* (SLO). Thus, SLA compliance auditing aims at verifying that these SLOs are met. This task must be *automated* in order to be *efficient* and to enable *real-time reactions* in case of an SLA violation.

In fact, specifying SLAs on IP-based networks becomes viable through network device instrumentations for Quality-of-Service (QoS) measurements, not only of transport but also of application services. However, application service SLAs still pose challenges to their compliance auditing, due to the *variety* and the potential *complexity* of SLOs. An example for a complex SLO is the following detail specification of service availability: “In most cases, service requests from authorised users will be accepted. If a re-

18th IFIP/IEEE International Workshop on Distributed Systems: Operation and Management (DSOM 2007), October 29-31, San José, California, U.S.A.

quest from an authorised user is rejected or not responded within 15 seconds, then the next request for this service from the same user will be accepted. However, this next request must be made within the next 5 minutes and 1 minute must have been elapsed since the rejected or unresponded request." Thus, an *expressive* specification language is beneficial to formally specify such complex relations among various events.

A useful auditing framework must allow for the *distribution of auditing load* to separate auditor instances. The time and memory required for auditing may increase only *linearly* with an increasing number of audit data. Moreover, the framework must be *reusable* and *easily* adaptable to audit any *complex* SLO. Hence, this paper presents a *scalable* and *highly reusable generic* framework, termed *AURIC* (***Au***diting F***r***amework for ***I***nternet Servi***c***es), which supports *secure inter-domain* interactions and provides all necessary core functionality to conduct *automatically* potentially *complex* audit tasks.

The remainder of this paper is organized as follows. Section 2 discusses related work. While Section 3 presents the AURIC architecture for SLA compliance auditing, Section 4 describes its prototypical implementation. An extensive evaluation of AURIC with respect to its scalability and reusability is presented in Section 5, which is followed by Section 6, where conclusions are drawn.

2 Related Work

Current approaches in SLA management address the *formal specification* of a complete SLA in a *specific area*, *e.g.*, network or web services, or concentrate on measurements of a pre-defined set of SLA parameters [1], [6], [8], [10], [12], [13]. Hence, to modify or to extend an existing solution, particularly a commercial product, for its application to an SLO with a different logic, a larger effort is needed than if the solution has been based on a generic framework like AURIC. Moreover, most approaches support only *simple SLO terms* and do not consider possible *inter-domain* auditing interactions and their *security* requirements. While [7] discusses all relevant details of related work, the following paragraphs summarize major issues only.

The Web Service Level Agreement (WSLA) Framework proposes a concept for SLA management including online monitoring of SLA violation and defines a language to specify SLAs [13]. However, it *focuses on web services and supports only simple SLO terms*. A condition in a WSLA's SLO is simply a logic expression with SLA parameters as variables. WSLA does not support conditional expressions for SLO specifications and the framework does not expect to process metered data consisting of more than one field, *e.g.*, <IPAddress, PacketLossRatio>. Since the timepoint at which the value of a measured metric is transferred is considered as the measurement timepoint, batch processing of measured data is not supported.

In the area of Grid services, Cremona [14] is an architecture and library for the creation and monitoring of WS-Agreements, whose specification is worked out by the Grid Resource Allocation and Agreement Protocol Working Group (GRAAP-WG) of the Global Grid Forum. Cremona supports the implementation of agreement management, however, SLO monitoring is considered application specific, thus, no support to its implementation is available, except an interface to retrieve monitoring results.

The Project TAPAS (Trusted and Quality-of-Service Aware Provision of Application Services) proposes SLAng, a language for expressing SLAs precisely [16]. SLAng is defined using an instance of the Meta-Object Facility model, and its violation semantics is defined using Object Constraint Language constraints. To reduce the possibility of disagreement over the amount of errors introduced by the mechanism for SLA violation detection, a contract checker is to be automatically generated by using the metamodel of the language and associated constraints as inputs for a generative programming tool [15]. However, this approach leads to performance problems. Thus, in order to eliminate various drawbacks mentioned above, this paper presents an architecture for SLA compliance auditing as described in the next section.

3 AURIC SLA Compliance Auditing Architecture

Based on the generic model and architecture for automated auditing [9], the AURIC architecture for SLA compliance auditing has been implemented, which covers three main functions: metering, accounting, and auditing, as depicted in Fig. 1.

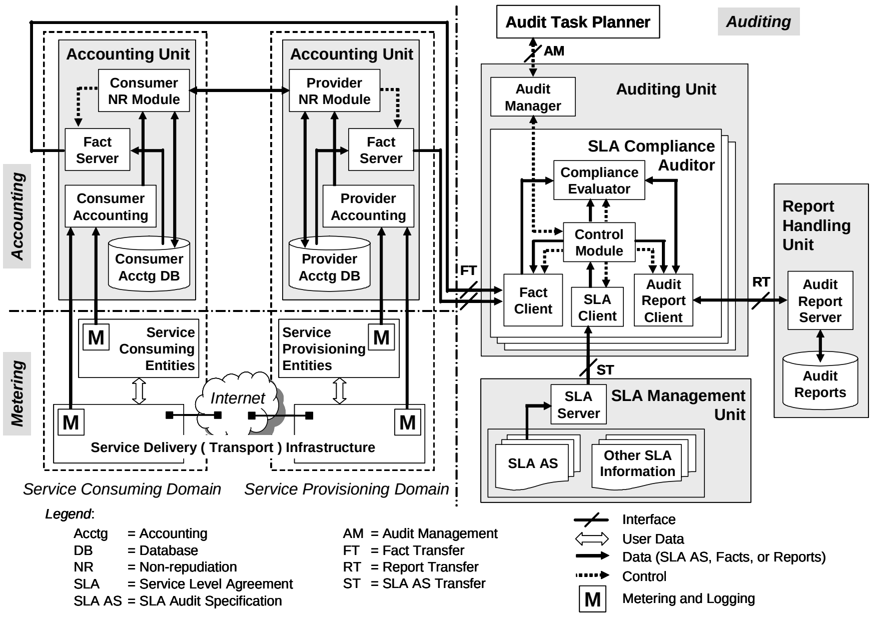


Fig. 1. AURIC SLA Compliance Auditing Architecture

Metering and Accounting: The quality level of a service being delivered must be metered to allow for the auditing of its SLA compliance. Metered data are collected and aggregated by accounting components to generate accounting data (termed Facts). Accounting data are passed to the non-repudiation (NR) module to generate evidence of service consumption. Generation and transfer of evidences require interactions between NR modules from both sides. Accounting data and evidences are stored in the accounting database and the respective Fact server is notified, so that they are transferred to the

SLA compliance auditor. If non-repudiation is not required, an NR module simply acts as a proxy between the accounting component and the database or the Fact server. The architecture and protocols for non-repudiation of service consumption supporting fairness and identity privacy in a mobile environment are available [7], [11].

Auditing: The main interactions between AURIC's components are for auditing. The auditing unit provides an auditing service through the Audit Management (AM) interface. The audit manager waits for audit requests and forwards each audit task received to an auditor. It also accepts requests relating to an audit task being conducted, *e.g.*, requests on its status and requests to stop an audit task. An audit task planner represents an entity which requests an auditing service from an auditing unit. An auditor retrieves data to be processed from various sources: accounting units, SLA management units, and Report handling units. Each of these components provides for a service to access its data through a data server component, namely a Fact server, an SLA server, and an Audit Report server respectively. Note that an Audit Report server also receives requests to store Audit Reports. All SLOs committed are assumed to be specified in a language, which allows for an automated auditing. The resulted specifications are called SLA Audit Specifications (SLA AS). Other SLA information, *e.g.*, user profile, service profile, are not relevant at this stage, and thus, are not explicitly listed in the figure.

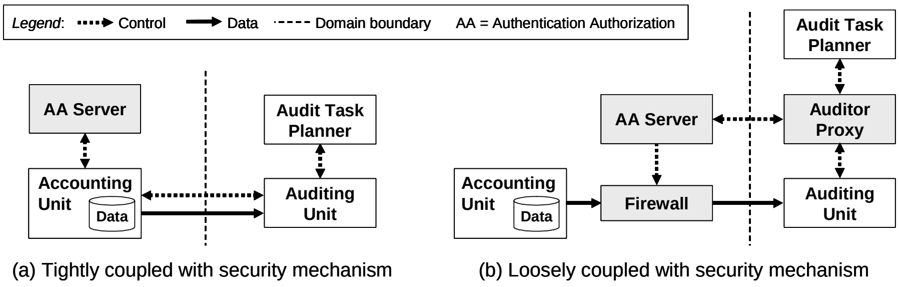
To communicate with various data servers, an auditor must contain the corresponding clients. The communication happens via the respective interface: SLA AS Transfer (ST), Fact Transfer (FT), or Report Transfer (RT) interface. The auditor must also contain a compliance evaluator to examine accounting data and Audit Reports based on the SLA AS obtained from the SLA client through the control module. The control module configures and controls other components in carrying out their functions. The Fact client retrieves accounting data and delivers them to the compliance evaluator. If needed, the Audit Report client retrieves and delivers Audit Reports to the compliance evaluator. Finally, this client sends Audit Reports obtained from the compliance evaluator to a Report handling unit. Table 1 briefly discusses suitable protocols for those interfaces.

Table 1. Auditing Interfaces

Interface	Description
AM	A new protocol for this interface is needed, however, following two communication patterns are sufficient to enable management interactions in normal and erroneous situations: Request-Answer and Notification pattern. A request message is used to initiate or terminate an audit task or to obtain its status information. An answer is sent as a response to a request message and it may contain error description, if any. A notification can be sent at any time to inform the respective audit task planner of completion of an audit task or any error occured during an audit.
ST	A URL is used to locate a particular SLA AS. Existing protocols such as HTTPS and SSH File Transfer Protocol are very well suited to be used to transfer SLA AS securely from an SLA manager to the auditing unit.
FT	For the purpose of transferring Facts, Diameter [2] protocol is very well suitable. The Base Accounting message pair is sufficient. However, to allow for selection of Facts a new Diameter command must be defined.
RT	Diameter is also suitable here, since the types of interactions are the same as for FT interface.

Security Considerations: Multi-domain support requires secure interactions and access control. Since in an SLA all parties involved are known in advance, security associations among those components can be established before interactions take place.

Having these security associations in place, authentication and authorization (AA) can be accomplished. As an example, suppose that accounting unit and auditing unit are located in different administrative domains. There are several ways of doing access control, *e.g.*, based on Authentication, Authorization, and Accounting (AAA) architecture [3]. In Fig. 2 (a), an AA server is contacted by the accounting unit to authenticate and authorize the auditing unit before it is allowed to send data to the auditing unit.



(a) Tightly coupled with security mechanism

(b) Loosely coupled with security mechanism

Fig. 2. Examples of Secured Access to Accounting Data

Access control can also be provided without intervening auditing functionality as shown in Fig. 2 (b). An auditor proxy is inserted between audit task planner and auditing unit. The proxy analyses audit tasks and requests access to the relevant accounting unit from the AA server of the respective domain. If there is a security association between the two domains, the access request is accepted and the firewall is configured to allow data flows between the auditing unit and the accounting unit. On receipt of a positive response from the AA server, the proxy forwards the audit task to the auditing unit. Finally, if necessary, a secure communication channel can be established to transfer data confidentially, based on security associations between those domains.

4 Implementation

Based on the proposed architecture, a prototypical implementation of an SLA compliance auditing framework in C++ is provided. The implementation aims at showing that developing an auditor can be done basically through specialization of a set of base classes to implement the SLO specific application logic. Fig. 3 depicts the implementation architecture of a *specific* SLA compliance auditor. The auditor is specific, since the application logic to audit a *specific SLO* is implemented as an integral part of the auditor. Thus, the auditor does not require an SLA client component to retrieve the SLA AS (cf. Fig. 1). However, various application logic corresponding to different SLOs can be implemented at compile time before one is chosen to be applied through a configuration file at run time. Thus, the need of a parser.

Each Fact and Audit Report is represented as a list of attribute-value-pairs (AVPs). Diameter [2] is chosen as the protocol for transferring Facts and Audit Reports due to its extensibility and the capability of its accounting message to carry a list of AVPs. Thus, the functionality of a Fact client and a Report client (cf. Fig. 1) is merged into a single entity called a Fact and Report client, which consists of a Fact and Report transfer

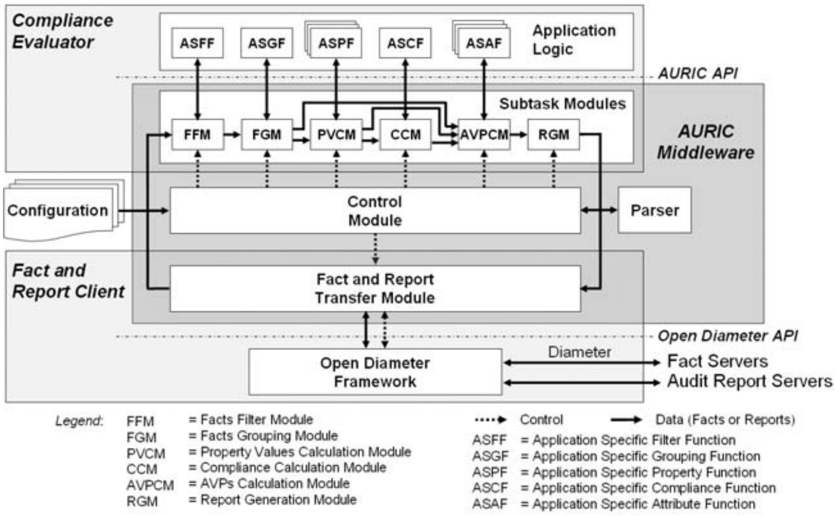

Compliance Evaluator
ASFF
ASGF
ASPF
ASCF
ASAF
Application Logic
AURIC API
Subtask Modules
FFM
FGM
PVCM
CCM
AVPCM
RGM
AURIC Middleware
Configuration
Control Module
Parser
Fact and Report Client
Fact and Report Transfer Module
Open Diameter API
Open Diameter Framework
Diameter
Fact Servers
Audit Report Servers
Legend:
FFM = Facts Filter Module
FGM = Facts Grouping Module
PVCM = Property Values Calculation Module
CCM = Compliance Calculation Module
AVPCM = AVPs Calculation Module
RGM = Report Generation Module
Control
Data (Facts or Reports)
ASFF = Application Specific Filter Function
ASGF = Application Specific Grouping Function
ASPF = Application Specific Property Function
ASCF = Application Specific Compliance Function
ASAF = Application Specific Attribute Function

Fig. 3. Implementation Architecture of an SLA Compliance Auditor

module implemented on top of the Open Diameter framework. The description of the Open Diameter implementation is given in [5]. Furthermore, to obtain a modular design, the author proposes to decompose an audit task into a sequence of subtasks:

1. ***Facts filtering***: Only Facts which are relevant for the SLO being audited are to be further processed. The filtered Facts are named *related Facts*.

2. ***Facts grouping***: Related Facts must be grouped, since they result from different service settings or observation periods. A group of Facts from a particular setting or period is named a *Fact-List*. A Fact-List being built is called an *open* Fact-List, whereas a Fact-List ready for auditing is called a *complete* Fact-List.

3. ***Property values calculation***: Each performance parameter of a service is characterized by a set of *properties*, whose values are calculated from the complete Fact-List examined in order to determine the compliance with the SLO.

4. ***Compliance calculation***: The *degree of compliance* with the SLO is calculated by applying the SLO specific compliance formula to the property values.

5. ***Report AVPs calculation***: The values for the report AVPs are calculated from various sources: the Fact-List, property values, and the compliance value.

6. ***Report generation***: As a result, a report is composed from the report AVPs.

Based on this decomposition, the compliance evaluator is developed, which consists of two parts: a sequence of subtask modules and a set of application logic. While the application logic implements SLO specific subtask functions, the subtask modules implement functionality which is common to all auditing applications, namely, management of Facts, Fact-Lists, and property values, as well as transfer of data between two subtask modules. The interface between a subtask module and its application logic is defined by the AURIC Application Programming Interface (API).

4.1 AURIC API

The auditing framework API provides five base classes to implement application logic (cf. Fig. 4). The parent class `SubtaskFunc` provides methods to parametrize the application specific subtask function derived, which are invoked by the auditing framework after the creation of the function based on the configuration file. Each base class offers a method `Process()`, whose purpose is described in Table 2 and which should be implemented by the developer of the auditing application.

Table 2. The Purpose of the API's `Process()` Methods.

Class	The Purpose of `Process()` Method
`Filter-Function`	To examine the accounting record encapsulated in the `Fact` object and return true or false to denote whether the record is related to the SLO being audited. A `Fact` object provides for methods to get information about the accounting record encapsulated in the object, *e.g.*, the value of a particular attribute.
`Grouping-Function`	To examine the accounting record encapsulated in the `Fact` object and assign the record to one or more Fact-Lists with the help of `OpenFactLists` object. An `OpenFactLists` object provides for methods to manipulate open Fact-Lists managed by the auditing framework, *e.g.*, to add a Fact into an open Fact-List and to close an open Fact-List.
`Property-Function`	To calculate a property value from the list of related accounting records encapsulated in the `FactList` object. A `FactList` object provides for methods to manipulate and to access information about accounting records encapsulated in the object, *e.g.*, the number of records, the sum of the value of a particular field of the records.
`Compliance-Function`	To calculate a compliance value from the list of property values encapsulated in the `PropertyValues` object. A `PropertyValues` object provides for methods to access property values.
`Attribute-Function`	To calculate a report attribute value from the list of related accounting records (encapsulated in `FactList` object), the list of property values (encapsulated in the `PropertyValues` object), and the compliance value.

4.2 Development of a General SLA Compliance Auditor

A *general* SLA compliance auditor is an auditor which can be used to audit *any* SLO without the need to modify and recompile the application logic. To implement a general SLA compliance auditor, following items must be available: an audit specification *language* to define in detail *how* an SLO is to be audited and an implementation of those five application specific classes as an *interpreter* of the audit specification language used. An audit specification language, named *Sapta*, has been developed [7].

A Sapta specification for auditing an SLO consists of a set of *function definition subspecifications* and a set of *function invocation subspecifications*. Each set of function definition subspecifications defines the application logic corresponding to those five functions defined in Section 4.1 to audit a specific SLO, whereas each set of function invocation subspecifications defines which function definition subspecifications are to be invoked and with which values for their parameters. The function invocation subspecifications in Sapta is usable as a configuration file for auditing, which consists of a `ComplianceCalculation` subspecification and a `ReportComposition` subspecification. Furthermore, the following principle is followed in the design of Sapta: The management (storage and transport) of Facts and Fact-Lists should be transparent to a pro-

```
class SubtaskFunc {
 public:
  virtual ~SubtaskFunc() {}
  virtual bool SetStringParam(
   unsigned int paramNo,
   const string& paramVal) {return false;}
  virtual bool SetNumberParam(
   unsigned int paramNo,
   float paramVal) {return false;}
  virtual bool SetBooleanParam(
   unsigned int paramNo,
   bool paramVal) {return false;}
};
class FilterFunction : public SubtaskFunc {
 public:
  virtual ~FilterFunction() {}
  virtual bool Process(
   const Fact& currentFact) = 0;
};
class GroupingFunction : public SubtaskFunc {
 public:
  virtual ~GroupingFunction() {}
  virtual void Process(const Fact& currFact,
   OpenFactLists& ofl) = 0;
};
```

```
class PropertyFunction : public SubtaskFunc {
 public:
  virtual ~PropertyFunction() {}
  virtual prop_value_t* Process(
   FactList& currentFactList) = 0;
};

class ComplianceFunction: public SubtaskFunc {
 public:
  virtual ~ComplianceFunction() {}
  virtual float Process(
   const PropertyValues& propertyValues) = 0;
};

class AttributeFunction : public SubtaskFunc {
 public:
  virtual ~AttributeFunction() {}
  virtual void Process(string& attrValue,
   FactList& currentFactList,
   const PropertyValues& propertyValues,
   float complianceValue) = 0;
};
```

Fig. 4. AURIC API

grammer of an audit specification. Accesses to and manipulations of Facts and Fact-Lists are to be supported through specific language constructs. Thus, in addition to conventional language constructs such as iteration and conditional branches, Sapta defines constructs which allow for a convenient specification of audit subtasks, *e.g.*, time schedule to evaluate completeness of a Fact-List (cf. Chapter 4 in [7] for further details).

5 Evaluation

The AURIC framework is evaluated with respect to its key requirements defined in Section 1. The scalability of the architecture is analyzed with respect to the number of SLOs, while the load scalability of its implementation, in terms of processing delay and memory requirements, is evaluated with respect to the number of Facts to be processed.

5.1 Scalability of Auditing Framework

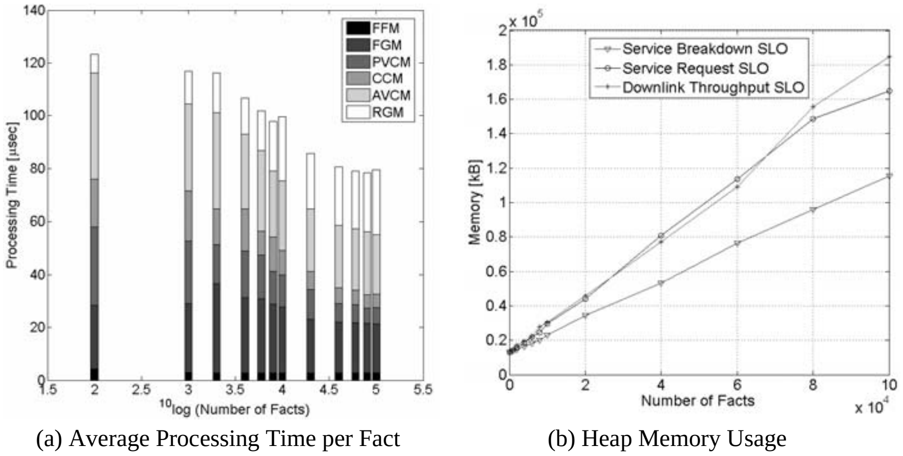
Suppose that there are *p* parties in a multi-domain environment and two SLAs are negotiated between any two parties (in one SLA a party takes the role of a service provider, in the other SLA the role of a customer). This full mesh relationship results in *p*(p-1)* SLAs. However, from the point of view of each party only *2*(p-1)* SLAs are relevant. Unlike other approaches which use an auditor instance per SLA, AURIC defines an auditor instance per SLO. The number of SLOs (n_{SLO}) does not depend on the number of SLAs (n_{SLA}), but on the number of services (n_{svc}). Assuming that each service has a maximum of *c* SLOs, then n_{SLO} is bound by $c*n_{svc}$. Table 3 compares the scalability of AURIC architecture with the other approaches, where n_A is the number of auditor instances required and $n_{A,max}$ is its upper bound. Although all approaches show a *linear scalability*, AURIC does have an *advantage* over the other: the number of services and SLOs grows much slower than the number of customers (SLAs).

With respect to the load scalability of an auditor, the number of Facts to be audited is crucial. There is a limit to the processing speed of an auditor, which determines the amount of Facts allowed per time unit. The amount of Facts can increase due to, *e.g.*, more sessions, which are generated. By *scaling up* the auditor, more Facts can be audited. However, this problem can also be solved by *scaling out* the auditor, since accounting data for the same SLO can be partitioned (*e.g.*, based on CustomerID) and delivered to several instances of auditors, all responsible for the same SLO.

Table 3. Scalability Comparison

Approach	n_A	$n_{A,max}$	**Order of** n_A
WSLA Framework, Cremona, TAPAS SLAng	n_{SLA}	$2 * (p-1)$	O(n)
AURIC	n_{SLO}	$c * n_{svc}$	O(n)

Auditor Processing Time: To evaluate the processing time, three SLO specific auditors are implemented based on the AURIC framework. Each auditor is responsible for auditing one of the three SLOs: Service Breakdown SLO, Service Request SLO, and Downlink Throughput SLO. The measurement of the processing time is done on a host with a Pentium 4 CPU 1.80 GHz, 512 MB main memory. Facts to be processed are delivered at once in a single batch to the auditor, and experiments are carried out with different numbers of Facts. In each experiment the time needed by those Facts to pass processes from the first to a certain subtask module is measured. Each experiment is run 10 times with the same configuration to obtain an average value of the processing time. For example, results show that it takes *in average* 7.94 s (with a *standard deviation* of 0.16 s) to process 100,000 Facts delivered at once through the sequence of *all* subtask modules in auditing the service breakdown SLO.



(a) Average Processing Time per Fact (b) Heap Memory Usage

Fig. 5. Load Scalability

Fig. 5 (a) depicts as an example the average processing time *per Fact* in *each* subtask module for auditing service breakdown SLO. Other use cases see similar results. The time required by an auditor to accomplish its task is determined by the total number of Facts to be processed, the number of *related* Facts after being *filtered*, the number of

Fact-Lists after being *grouped*, and the *complexity* of the SLO defined. In all use cases, for a large number of Facts the processing time per Fact in each subtask module exhibits a relative constant value as expected. Thus, AURIC shows a scalable implementation. **Auditor Heap Memory Usage:** Memory requirements of the auditor are important, especially in relation to the number of Facts. Hence, for those three use cases the memory usage is obtained from /proc files [4]. The virtual memory usage of the heap determines the dominating aspect, thus, all other memory usage is omitted. If all Facts are delivered *at once* to the auditor, a *linear* increase of heap memory usage with an increasing number of Facts is expected, since more memory will be needed to store more Facts. This behavior is shown in Fig. 5 (b), showing that the AURIC implementation scales.

5.2 Reusability of Auditing Framework

High reusability is a very important property to be fulfilled by an auditing framework. AURIC's reusability is shown by demonstrating that most of the auditing components do not need to be adapted or replaced, when developing a new auditing application based on the framework. Assuming the example of the following application logic to determine compliances of Facts with a certain SLO:

- If a `Fact` belongs to the SLO to be audited then `ff1(Fact)` is true.
- The value of `gf1(Fact, OpenFactLists)` identifies the `FactList` to which the `Fact` belongs (*e.g.*, all accounting records about (un)availability of service S within a month are to be grouped in order to decide on SLO compliance). If a `FactList` is complete, then `gf2(FactList)` is true.
- A `FactList` complies with the SLO if the value of `cf1(pf1(FactList), pf2(FactList))` is 1 (*e.g.*, if service S may down at most 3 times which are longer than 5 minutes, and the total downtime may not exceed 30 minutes, then `pf1()` would count the number of breakdowns longer than 5 minutes and `pf2()` would calculate the total downtime).
- If a `FactList` does not comply with the SLO a report consisting of `pf1(FactList)`, `pf2(FactList)`, `af1(FactList)`, and `cf1(pf1(FactList), pf2(FactList))` is to be generated.

This logic is easily implemented into AURIC by writing those five application-specific functions. Fig. 6 depicts the *simplified* code snippets. Having defined these subclasses, the programming job is done and an executable auditor for this specific SLO can be compiled. All other functionality is provided automatically by the framework, *e.g.*, interactions with Fact/Report servers to obtain Facts and to deliver Audit Reports, management of Facts, Fact-Lists, property values, and execution of methods invoked by audit subtasks, as well as transfer of data between audit subtasks.

Before invoking the newly developed auditor, a configuration file written in Sapta needs to be created. The framework consults this file to determine, which subclasses are to be used by each audit subtasks and to determine the composition of an Audit Report. For the example above, the content of the configuration file is shown in Fig. 7. Furthermore, it is likely that several SLOs share the same application logic for specific functions, *e.g.*, a `PropertyFunction` to determine the average value of a certain field in the accounting records. This subclass needs to be coded once and can be used for various SLOs through auditor configurations. Thus, the framework also supports reuse of application logic without code duplication in addition to the reuse of its own components.

```
class FF_SLO1 : public FilterFunction {
 public:
  bool Process(const Fact& currentFact)
    {return ff_1(currentFact);}
};
class GF_SLO1 : public GroupingFunction {
 public:
  void Process(const Fact& currentFact,
               OpenFactLists& ofl) {
    thisFactListId = gf_1(currentFact, ofl);
    ofl.Assign(thisFactListId, currentFact);
    if (gf_2(ofl.GetFactList(thisFactListId)))
     {ofl.CloseFactList(thisFactListId);}
  }
};
class PV_SLO1 : public prop_value_t {
// define variables to store a property value
};
class PF_1_SLO1 : public PropertyFunction {
 public:
  prop_value_t* Process(FactList& currFL) {
    PV_SLO1* pv = new PV_SLO1;
    // assign pf_1(currFL) to variables in pv
    return ((prop_value_t*)pv);
  }
};
```

```
class PF_2_SLO1 : public PropertyFunction {
 public:
  prop_value_t* Process(FactList& currFL) {
    PV_SLO1* pv = new PV_SLO1;
    // assign pf_2(currFL) to variables in pv
    return ((prop_value_t*)pv);
  }
};
class CF_SLO1 : public ComplianceFunction {
 public:
  float Process(const PropertyValues& pVal) {
    PV_SLO1& pv1 = (PV_SLO1&)
      pVal.GetPropertyValue(1);
    PV_SLO1& pv2 = (PV_SLO1&)
      pVal.GetPropertyValue(2);
    return (cf_1(pv1, pv2));
  }
};
class AF_SLO1 : public AttributeFunction {
 public:
  void Process(string& attrValue,
      FactList& currentFactList,
      const PropertyValues& propertyValues,
      float complianceValue) {
    attrValue = af_1(currentFactList);
  }
};
```

Fig. 6. Deriving Application Specific Functions

```
ComplianceCalculation CC_SLO1 {
        FF_SLO1
        >> GF_SLO1
        >> PF_1_SLO1, PF_2_SLO1
        >> CF_SLO1
}
```

```
ReportComposition RC_SLO1 {
        [Field1 eq GF_SLO1 >> AF_SLO1],
        [Field2 eq PF_1_SLO1],
        [Field3 eq PF_2_SLO1],
        [Field4 eq CF_SLO1]
}
```

Fig. 7. Example Configuration in Sapta

6 Summary and Conclusions

Existing approaches in SLA compliance auditing lack a *general applicability* and concentrate on formal specifications of SLAs rather than on the auditing of SLOs. These pure specification approaches lead to the potential unawareness of system designers on how manifold and complex an SLO for application services can be beyond a guarantee of traditional QoS parameters. Thus, AURIC has been designed based on a *generic model and architecture*. Since the architecture neither assumes specific services nor specific SLOs, it is *general* and applicable to the full range of Internet service types. Furthermore, AURIC architecture is shown to be *linearly scalable* with respect to the number of SLOs due to the possibility to employ an auditor per SLO and to divide the load. The framework implementation also shows a linear scalability of the processing time and memory usage with respect to the number of Facts to be audited.

AURIC framework's functionality is *highly reusable*, which is achieved through the functional decomposition of an audit task into a sequence of subtasks to allow for a modular specification, and through the separation of common audit functionality from SLO-specific auditing logic, as well as a formal language *Sapta* to specify complex audit tasks in full detail. The framework implements the required common audit functionality and offers an API to implement the application logic for auditing a specific SLO. Using AURIC framework, a developer does not need to be concerned about the control of data flow, management of audit data, and data transport. Therefore, the efforts to develop an auditing application based on AURIC framework are largely reduced.

Acknowledgments

The work has been performed partially in the framework of the EU IST Project Akogrimo (IST-2004-004293), the EU IST Project Daidalos II (IST-2005-026943), and the EU IST Network of Excellence EMANICS (IST-NoE-026854).

References

1. Agilent Technologies: *Measuring Web Quality of Service with the New HTTP Test in Fire-Hunter 4.0*; White Paper, Agilent Technologies Inc., November 2002.
2. P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko: *Diameter Base Protocol*; IETF, RFC 3588, September 2003.
3. C. de Laat, G. Gross, L. Gommans, J. Vollbrecht, D. Spence: *Generic AAA Architecture*; IETF, RFC 2903, August 2000.
4. S. Duddi: *Demystifying Footprint*; mozilla.org, March 2002.
5. V. I. Fajardo: *Open Diameter Software Architecture*; 2004.
6. G-NE GmbH: *Konzeptionsansatz: Qualitätssicherung in IT-Outsourcing-Projekten mittels einer unabhängigen Prüfinstanz*; Confidential Document, 2002.
7. Hasan: *A Generic Auditing Framework for Compliance Verification of Internet Service Level Agreements*; ETH Zürich, Switzerland, PhD Thesis, Shaker Verlag GmbH, Aachen, 2007.
8. Hasan (Editor): *A4C Framework Design Specification*; Deliverable D341, Sixth European Union Framework Programme, IST Project “Daidalos”, September 2004.
9. Hasan, B. Stiller: *A Generic Model and Architecture for Automated Auditing*; 16th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM’05), Barcelona, Spain, October 24-26, 2005.
10. Hasan, B. Stiller: *Auditing Architecture for SLA Violation Detection in QoS-Supporting Mobile Internet*; IST Mobile and Wireless Comm. Summit 2003, Aveiro, Portugal, June 2003.
11. Hasan, B. Stiller: *Non-repudiation of Consumption of Mobile Internet Services with Privacy Support*; IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob’05), Montreal, Canada, 2005.
12. Itellix Software: *Wisiba*; Datasheet, 2003.
13. A. Keller, H. Ludwig: *The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services*; Journal of Network and Systems Management, Vol. 11, Issue 1, pp. 57-81, March 2003.
14. H. Ludwig, A. Dan, R. Kearney: *Cremona: An Architecture and Library for Creation and Monitoring of WS-Agreements*; International Conference on Service Oriented Computing, New York, USA, November 2004.
15. J. Skene, W. Emmerich: *Generating a Contract Checker for an SLA Language*; EDOC 2004 Workshop on Contract Architectures and Languages, Monterey, California, 2004.
16. J. Skene, D. Davide Lamanna, W. Emmerich: *Precise Service Level Agreements*; 26th International Conference on Software Engineering, Edinburgh, UK, May 2004.
17. Telemanagement Forum: *SLA Management Handbook*; V1.5. GB917, 2001.
18. D. C. Verma: *Service Level Agreements on IP Networks*; Proceedings of the IEEE, Vol. 92, Issue 9, September 2004.

LINUBIA: A Linux-supported User-based IP Accounting

Cristian Morariu, Manuel Feier, Burkhard Stiller

Department of Informatics IFI, University of Zurich, Switzerland.
[morariu|stiller]@ifi.uzh.ch

Abstract. Obtaining information about the usage of network ressources by individual users forms the basis for establishing network billing systems or network management operations. While there are already widely used accounting techniques available for measuring IP network traffic on a per-host basis, there is no adequate solution for accounting per-user network activities on a multi-user operating system. This work provides a survey on existing aproaches to this problem and identifies requirements for a user-based IP accounting module. It develops a suitable software architecture LINUBIA and proposes a prototypical implementation for the Linux 2.6 operating system, which is capable of providing per-user accounting for both the IPv4 and the IPv6 protocol.

1 Introduction and Problem Statement

The Internet is rapidly growing and fundamentally influencing economic, cultural, and social developments worldwide. While more and more people take advantage of this global network and new services are being deployed every day, more network resources are consumed. This creates the need for effective accounting mechanisms that are closely coupled to authentication mechanisms, *e.g.,* in support of network management tasks, charging requirements, or intrusion detection systems for systems and users. Often it becomes necessary to know what amount and which type of network traffic a specific network user is generating.

Today, as networking is moving towards an all-IP network [9] an accounting system integrated into the IP layer seems the most straight forward solution. This approach allows for the same accounting mechanisms to be used regardless of the application and the transport protocol carried over IP, or the data link layer and physical connection the IP runs on top of.

Although the accounting of IP network traffic has received wide attention since the beginning of the Internet [18], existing systems have a major drawback by looking strictly to the IP packet captured on the wire. Such an approach allows for the mapping of each IP packet to an end-sytstem, which sends or receives the packet, but it is unable to specify, which user was responsible for generating the traffic. Multi-user operating systems often use a single IP address, which is shared among different individual users. Since multiple users may be connected remotely at the same time to the same machine and may have different applications that generate IP traffic being transported over the network, it is impossible to identify how much traffic each of these users generated by just looking into the IP traffic at the router level.

18th IFIP/IEEE International Workshop on Distributed Systems: Operation and Management (DSOM 2007), October 29-31, San José, California, U.S.A.
Best Student Paper Award

Therefore, this paper proposes the user-based IP accounting architecture named LINUBIA, which uses a Linux kernel extension and a library for accessing this extension, for mapping each IP packet sent or received to the responsible user. This solution allows for splitting network costs in case of usage-based charging or may allow detection of the user or process that was responsible for illegal IP traffic.

Fig. 1 depicts a typical scenario for using the new user-based IP accounting infrastructure. In an enterprise network users are typically authenticated by using a centralized authentication server such as LDAP (Light-weight Directory Access Protocol) [15] or Kerberos [12] and they may access the network from any terminal or working station that is configured to use the central authentication server. Upon authentication the device to which the user logged on to starts to meter the network usage and sends periodic accounting records to the accounting server. Since the network usage is mapped to user identifiers (ID) and a user uses the same ID with any device he is allowed to connect to, the accounting server may aggregate the network usage from different devices within the network and present users with detailed and aggregated information about network traffic they created.

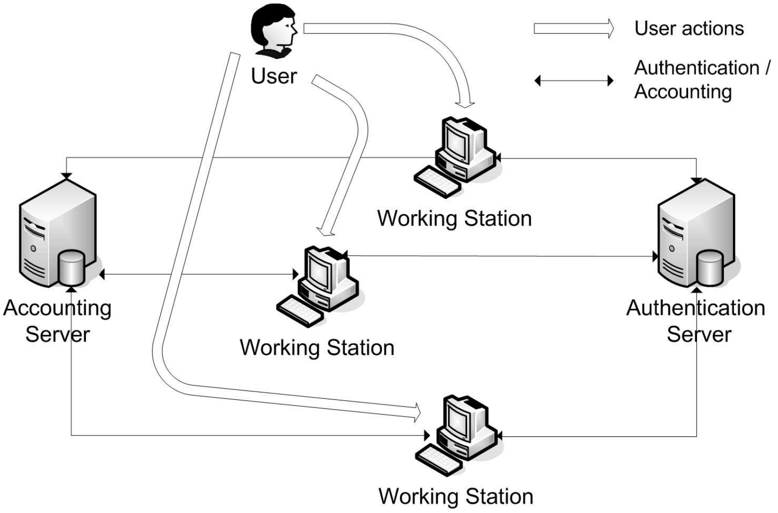


Fig. 1. Enterprise Scenario

The remainder of the paper is structured as follows. Section 2 presents a brief overview on related work in the field of IP accounting followed by Section 3, which outlines the design of the new approach. Section 4 gives major implementation details. While Section 5 discusses evaluation results, finally, Section 6 draws conclusions and povides ideas about future work.

2 Related Work

Some of the first guidelines for measuring internet traffic have been proposed in [18]. The IETF defined a generic AAA architecture (Authentication, Authorization, and Accounting) [10] that formed the basis for RADIUS and [14] Diameter [2] protocols.

The AAA architecture and its related protocols do not define how measurements need to be done, but specify how to transport and collect the data measured. SNMP (Simple Network Management Protocol) [3] allows network administrators to monitor and manage network traffic performance. Each network device in a SNMP managed network has a management agent that resides on that device. The management agent collects information from different counters and configuration parameters of the managed device and makes them available to a Network Monitoring Station (NMS) via the SNMP protocol. The information collected via SNMP is typically highly aggregated (*e.g.*, summary of data transferred on an interface or average data rate during the last n seconds). RTFM (Real-time Flow Monitoring) [1] and RMON (Remote Monitoring) [19] also use SNMP for transporting measured data. None of these solution offers any support for user-based IP accounting.

Commercial accounting systems, such as NetFlow [6], NARUS [11], or Juniper's J-Flow Accounting [8], lack the support for accounting on a per-user basis. The authors of [4] propose a method for accounting TCP connections on a per-user basis. Their solution is based on introducing an additional step in the TCP connection set-up phase for checking the authenticity of the user. If a TCP session is started all the traffic is reported to the user, who started the session. [20] proposes to use multiple IP addresses on a multi-user devices and use a distinct IP address for every user. This would allow traditional IP traffic accounting systems to be used for user-based IP accounting. NIPON [13] and [21] introduce an agent-based solution, where an agent is set up on a host with multiple users; this agent is designed to collect required traffic information directly on that host, but without having to change the operating system (kernel), so that it can also be used with closed source systems, like Solaris or Windows. The solution is based on capturing all trafic on the network interface to identify local traffic and to correlate it to local users. The drawback of this solution is the need to monitor all traffic on the link in case of shared links such as Fast Ethernet.

The UTAdragon project (Useful Traffic Accounting Dragon) [17] retrieves network data by collecting network and process information using the `/proc/net` interface. Data about open network connections and processes that use them are collected and recombined to create a table showing, which system user has consumed how much network traffic. The accounting data is stored in a MySQL Database, allowing further processing or aggregation. UserIPAcct (User IP Accounting) [16] is an extension to the Linux kernel originally developed in 1994. The development has stopped in 2000 and the latest beta version available addresses kernel 2.2. This system extends the Linux kernel in a way that it becomes able to attribute IP traffic to local system users. This code is not compatible with modern Linux kernels (*e.g.*, 2.6) and also does not support IPv6.

Comparing to existing work, LINUBIA proposes a user-based IP accounting system embedded into the latest generation of the Linux kernel (v2.6) and capable of performing accounting for IPv4 as well as IPv6 traffic. Moreover, LINUBIA reports measured traffic separated on different transport protocols. An important difference to previous approaches is having not only a Linux kernel extension for user-based IP accounting, but a solution that can easily integrate into existing authentication and accounting systems by using standard protocols such as LDAP and Diameter. Having the accounting module embedded into the Linux kernel enables, besides traffic accounting, later extensions to perform IP traffic access control based on users or applications which generated the traffic. The solution proposed here follows an architecture close to the one proposed in [16].

3 Design

Based on the following use cases a summary of the main requirements for the new user-based IP accounting architecture, termed LINUBIA is derived. Based on these requirements that have been identified the design of the proposed solution is detailed.

3.1 Motivating Use Cases

Network Traffic Billing System

The first scenario deals with the case of a grid infrastructure spanning across a larger area on top of which customers may run their own grid applications. A grid user will typically install its applications on multiple nodes and these run typically with the user’s privileges. The grid operator may use the user-based accounting module in order to split network costs (traffic created by grid applications is typically high) among all customers based on the amount of traffic they created.

Individual Load Monitoring and Abuse Detection

The second scenario addresses the case of an institution, for example a university, which offers its students the possibility to use the Web for research and communication purposes, but does not want them to excessively waste precious network bandwidth for sharing videos, filesharing, and the like. The system setup is done in a way that a student can log into one of many computers at the university with his personal credentials. The user account information is stored in a centralized LDAP directory, so a specific student has uses the same user ID (UID) in every system he logs into. A script can regularly copy usage information to a database server, where it is stored and accumulated with the traffic footprint of other users in order to detect possible anomalies in the traffic under investigation. The system administrator has the possibility to monitor network usage of students, independent of applications or the computer they use. With the help of this information he can detect and quantify abuses, suspend accounts of the respective users, or initiate further investigations.

Service Load Measuring

The third scenario handles the identification of applications, which generate abnormal traffic. For example, on a Linux server different services may be operational, some of them may not be using well-known ports (*e.g.*, a bit-torrent client, which constantly changes ports it is running on). On that router connecting this server to the Internet, the administrator can monitor how much traffic this server created, but he can only identify applications based on port numbers. In case of applications that change these ports the use a user-based IP accounting module eases traffic monitoring for these type of applications.

3.2 Requirements

Based on these use cases above as well as a general observation of achieving a practical and implementable solution the following four key requirements for IP accounting LINUBIA have been identified:

- The IP accounting module shall account for IPv4 and IPv6 traffic information on a per-user bases operating the Linux operating system.
- The IP accounting module shall allow for application-based traffic accounting.
- An API interface shall be available for configuring the IP accounting module and retrieving accounted for data.
- The performance impact of the IP accounting module on the networking subsystem should be kept minimal.

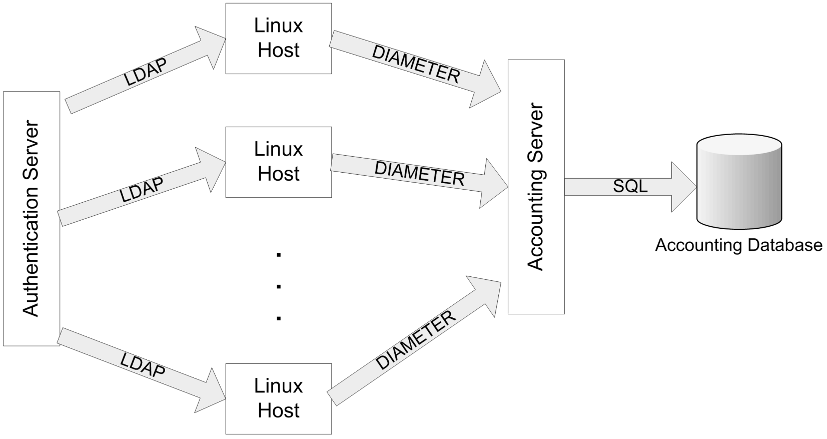


Fig. 2. Network Architecture

3.3 LINUBIA Architecture

The architecture of an enterprise network having LINUBIA running on the Linux end-hosts, consists of both a network architecture (cf. Fig. 2) that defines the network components required for LINUBIA and a end-host architecture (cf. Fig. 3) that defines the software components required within an end-system to support LINUBIA.

Two types of devices may be identified: regular Linux hosts, which may be used by users and the accounting domain infrastructure (consisting of an authentication server, a data aggregation server, and a storage server). Linux nodes use an authentication server for veryfying user credentials. Whenever a user logs in to a Linux host all the processes started by the user will run with the global UID of the user. Each Linux host has LINUBIA enabled. Accounted for data is encapsulated in accounting records and it is transported from each Linux host to an accounting server using the Diameter protocol. The accounting server further stores the accounting records in a central database. For supporting this an accounting client runs on each host, collects the data accounted by LINUBIA and sends it to an accounting server using the Diameter protocol.

3.3.1 End-Host Accounting Architecture

This section shall describe in detail the different components and their interractions required within an end-host in order to support user-based IP accounting. Regular operating systems do not offer a function to autonomously measure user-specific IP traffic. Therefore, a host needs to be modified in order to be able to perform such a task. Fig. 3 shows how this can be achieved by modifying the Linux operating system kernel, which resides between the networking hardware and applications in the user space. The kernel allows network applications to access the TCP/IP stack via the network socket interface; it contains routines to send outgoing IP packets to the network and deliver incoming packets to the destination applications. These routines and the kernel have to be extended in order to measure, store and export the desired accounting information associated with each accounting-relevant IP network operation. This is done by a kernel accounting extension that consists of a number of components which are added to the kernel.

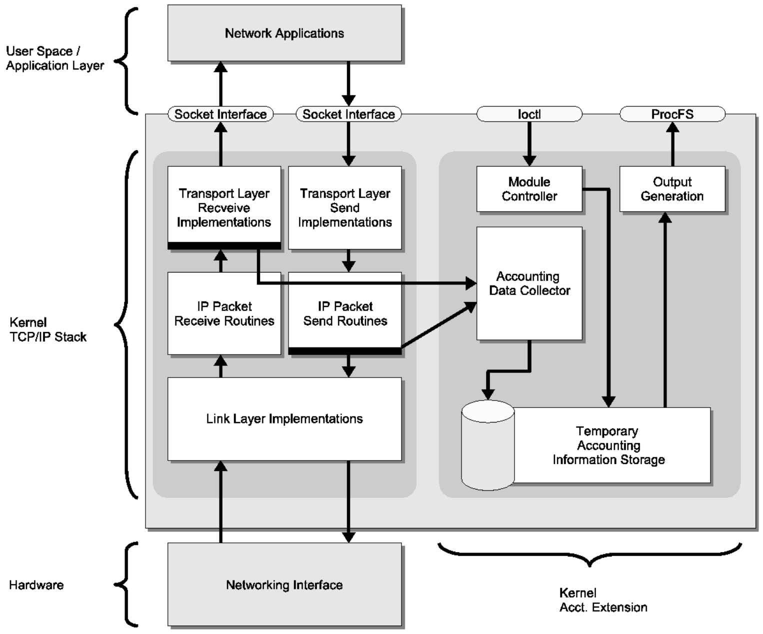


Fig. 3. End-Host Architecture

The *information storage* component is responsible for the temporary storage of accounting information collected. The *data collector* component retrieves the necessary information from the IP networking subsystem and puts it in the storage component. The *output generation* component reformats the internal data before exporting it to user space via the proc filesystem (`procfs`). The *module controller* provides facilities to manage records stored, for example to reset all records of a specific user. It uses the `ioctl` interface.

This architecture is designed to extract and export user-specific IP accounting information from the kernel to user space for further processing. The data is stored temporarily in the main memory by the kernel module. Data aggregation and persistent storage are done outside the kernel. in order to keep the load on the kernel as low as possible.

3.3.2 Integrated View

Fig. 4 shows the integration of the host specific architecture into the network architecture. In addition to the kernel-based accounting architecture sketched in 3.3.1 two aditional components are required for building accounting applications on top of LINBUIA. The first component is an *accounting library* that provides the API for querying and configuring the accounting module. It enables applications to access the kernel interfaces of the accounting extension.

The second component is a Diameter accounting client that uses this library to fetch the user-based IP accounting records from the kernel and sends them to a remote data aggregation server using the Diameter protocol. The aggregation server can evaluate and store the accounting data persistently, for example by using a separate database server.

A flexible system authentication back-end and Name Service Switch (NSS) configuration allows that a unique user account of a centralized user database (on a remote directory) can be used on any user host; the suggested interface being used for this is LDAP. The intention is that multiple hosts use the same user database and therefore the same UIDs for individual users, making users and associated accounting records uniquely identifiable across distinct hosts.

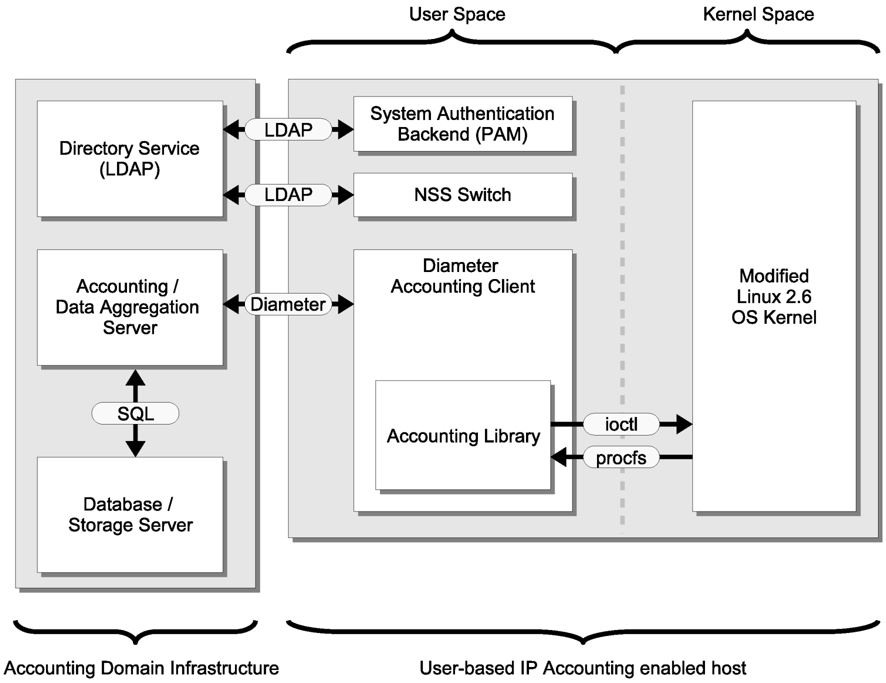

User Space
Kernel Space
Directory Service (LDAP)
LDAP
System Authentication Backend (PAM)
LDAP
NSS Switch
Accounting / Data Aggregation Server
Diameter
Diameter Accounting Client
Modified Linux 2.6 OS Kernel
SQL
Accounting Library
ioctl
procfs
Database / Storage Server
Accounting Domain Infrastructure
User-based IP Accounting enabled host

Fig. 4. Integrated View

4 Implementation

The implementation of the host-based extension is based on the code layout of the useripacct project [16] and is entirely written in the C programming language [5]. Compared to the other investigated approaches, LINUBIA supports 64 bit counters, provides real-time traffic statistics and allows parallel accounting of IPv4 as well as IPv6. The accounting system was implemented for modern 2.6 series Linux kernels and supports both IPv4 and IPv6.

The information triplet to be extracted from each IP network operation consists of the IP packet size, the packet owner (user), and the network and transport protocols involved with the operation. Unfortunately, the required routines and protocol headers are distinct for IPv4 and IPv6, and for incoming traffic, the information cannot be retrieved at the IP layer, like it is the case for outgoing traffic. This required the embedding of the accounting module routines in the transport layer implementation. A shortcoming of this approach is a scatter of the LINUBIA code across several files in the Linux kernel network subsystem.

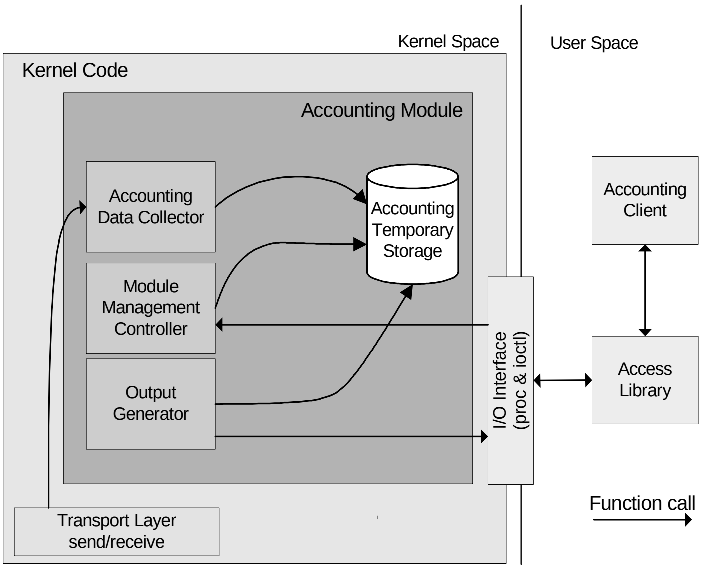

Kernel Space
User Space
Kernel Code
Accounting Module
Accounting Data Collector
Accounting Temporary Storage
Module Management Controller
Output Generator
Accounting Client
Access Library
I/O Interface (proc & ioctl)
Function call
Transport Layer send/receive

Fig. 5. Implementation Architecture

The *data collector* can extract the size of a packet from IP packet headers; the sum of the transferred IP packet sizes equals the IP traffic. The network and transport protocol types can be determined by identifying the kind of the network routine or by also inspecting the IP packet header. The user information can be determined by looking up the ownership properties of the network socket corresponding to a packet. As it is possible that IP packets are sent or received that have no associated local network socket, there are rare situations where traffic cannot be attributed to a regular user. This is handled by directing such accounting information to the record of a special user “nobody”.

The *information storage* component is implemented as a number of records that are connected in groups of doubly-linked lists. Each record contains the UID as the primary identification attribute as well as the measured IP traffic values for different network and transport protocols. Users are dynamically added when they start using IP-based networking.

Upon request, the *output generation* component loops through these lists to create a table with all users and their traffic records which is exported to the proc file system. The user space library reads a special item in the `proc` filesystem that is exported by the kernel extension and contains the temporary accounting information. The library recreates the record structures so that they can be easily accessed by other applications, such as the accounting client. It also provides functions to send commands to the *module controller*, using the `ioctl` interface. The accounting client sends locally detected accounting records to the accounting server using the Diameter protocol. Within Diameter, records are structured as sets of (predefined) Attribute-Value Pairs (AVP). The sample accounting client and sample server communicate in regular intervals by using accounting sessions, where an accounting session contains current records for one user, as delivered by the accounting library. Besides the accounting Attribute-Value-Pairs (AVPs) proposed in [2], a set of parameters have been defined as shown in Tab. 1.

Table 1. New AVPs for Linux User-Based IP Accounting

AVP Name	AVP Code	AVP Name	AVP Code
Linux-Input-IPV4-Octets	5001	Linux-Input-IPV4-TCP-Octets	5101
Linux-Output-IPV4-Octets	5002	Linux-Output-IPV4-TCP-Octets	5102
Linux-Input-IPV6-Octets	5003	Linux-Input-IPV4-UDP-Octets	5103
Linux-Output-IPV6-Octets	5004	Linux-Output-IPV4-UDP-Octets	5104
Linux-Input-TCP-Octets	5005	Linux-Input-IPV6-TCP-Octets	5105
Linux-Output-TCP-Octets	5006	Linux-Output-IPV6-TCP-Octets	5106
Linux-Input-UDP-Octets	5007	Linux-Input-IPV6-UDP-Octets	5107
Linux-Output-UDP-Octets	5008	Linux-Output-IPV6-UDP-Octets	5108

5 Evaluation

The evaluation of the user-based IP accounting module was performed both in terms of functional and performance evaluation. The tests have shown that the requirements described in Sect. 3.2 have been fully met. The set of experiments that have been performed in order to test the functionality, accuracy and performance of the accounting module used a network set-up as the one described in Fig. 6. The testing environment consists of two hosts that are connected in a LAN by a Fast Ethernet switch as seen in the figure. Both hosts run a Linux 2.6 operating system and use IPv4 as well as IPv6. Both hosts have Fast Ethernet network adapters.

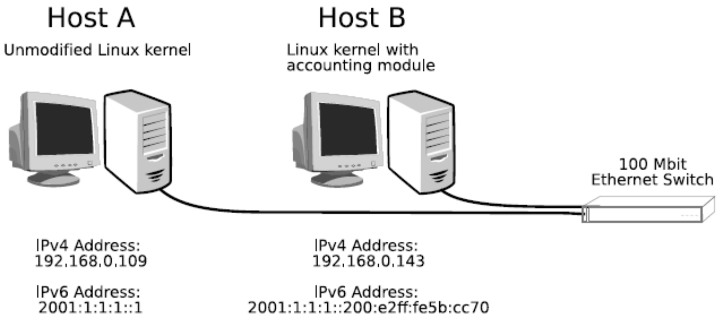

Host A
Unmodified Linux kernel
Host B
Linux kernel with accounting module
100 Mbit Ethernet Switch
IPv4 Address: 192.168.0.109
IPv6 Address: 2001:1:1:1::1
IPv4 Address: 192.168.0.143
IPv6 Address: 2001:1:1:1::200:e2ff:fe5b:cc70

Fig. 6. Testing environment

For testing the accuracy of the accounting module several tests have been performed in which TCP, UDP, and ICMP incoming and outgoing IPv4 and IPv6 traffic was generated and accounted for. The experiments have shown that the accounting module correctly accounts for IP traffic. During experiments it was observed that some traffic cannot be mapped to any user (such as scanning traffic or incoming ICMP messages). Such traffic is accounted for the system user by the accounting module. Another observation concerns ICMP traffic that appears to be exclusively mapped to the system user and not to the user who actually sent the message. The reason for this is that raw socket operations are considered critical and only possible for user root, also for security reasons (a regular user can only execute the ping program because it has the *SUID-bit* set, thus being executed under root context).

Tab. 2 shows the results of a first test consisting of a 256 MB file transfer over a Fast Ethenet link with and without LINUBIA using IPv4 and IPv6. The purpose of this test was to identify the impact of accounting on the performance of the Linux network subsystem. As the table shows there is only a small impact (0.83% for IPv4 and 0.41% for IPv6) on performance observed when running with LINUBIA enabled.

Table 2. Average time for a 256 MB file transfer over a Fast Ethernet connection with and without the user-based IP accounting enabled (average numbers of 20 runs)

	Unmodified IPv4	Accounting IPv4	Unmodified IPv6	Accounting IPv6
Average time	21.815 s	21.998 s	22.102 s	22.193 s
Std. deviation	0.062 s	0.208 s	0.010 s	0.204 s

In Tab. 3 observed and estimated maximum throughput on a Linux box with and without LINUBIA are shown. For estimating the maximum throughput the Iperf [7] tool was used. The test with Iperf affirms that the measuring results are correct. Although the values are not totally equal, the dimensions are the same and the performance loss is marginal.

During the evaluation phase of LINUBIA the architecture and its implementation have been tested to check the functionality they provide and the performance impact on the Linux kernel network subsystem. These tests have shown that LINUBIA delivers the required accounting results, especially a per-user network activities result on a multi-user operating system, while having a small impact on the performance of the end-system under investigation.

Table 3. Average maximum throughput observed and calculated by Iperf over an Fast Ethernec connection, with and without the user-based IP accounting module enabled.

	Unmod. IPv4	Acct. IPv4	Rel. diff. (%)	Unmod. IPv6	Acct IPv6	Rel. diff. (%)
Manual (Mbps)	93.880	93.099	0.839	92.661	92.281	0.412
Iperf (Mbps)	94.080	91.700	2.595	92.880	92.870	0.012

6 Conclusions and Future Work

This paper demonstrates by a design and prototypical implementation that a user-based IP accounting approach is technically possible on modern Linux (2.6 series) operating systems. Additionally, it can be used also in the same version with the upcoming IPv6 network protocol and it can be integrated into an existing accounting infrastructure, such as Diameter. On one hand, users are not supposed to have only one computer device of their own (not to mention sharing one device with other users), but rather to have several devices for different purposes. On the other hand, the more computers become commodities for daily life and will be used by different people (producing networking-related and other costs), the more important it becomes to establish accounting systems, which offer a clear and secure user identification on the end-device and will probably have an integrated character. The current implementation shows a clear proof of concept.

Improvements are possible, *e.g.*, with the storage component, which can be done with a smaller memory footprint and also more efficiently by utilizing advanced data structures that will help to optimize access times. Another interesting issue determines the linkage of the networking subsystem to the socket interface, which also implies a link to the process management of the operating system. An advanced accounting module can offer IP accounting not only per user, but also per process. This allows for the identification, the management, or schedulability of processes not only by their CPU usage or memory consumption, but also by their network resource consumption. Finally, this leads to the creation of network filters or firewalls that allow for or deny network access to specific applications or users running on a host, instead of only allowing or denying specific services.

Acknowledgment.
This work has been performed partially in the framework of the EU IST Project EC-GIN (FP6-2006-IST-045256) as well as of the EU IST NoE EMANICS (FP6-2004-IST-026854).

References

1. N. Brownlee, C. Mills, G. Ruth, *Traffic Flow Measurement: Architecture*; RFC 2722, October 1999.
2. P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko, *Diameter Base Protocol;* RFC 3588; September 2003.
3. J. Case, M. Fedor, M. Schoffstall, J. Davin, *A Simple Network Management Protocol (SNMP)*; RFC 1157, May 1990.
4. R. J. Edell, N. McKeown, P. P. Varaiya: *Billing Users and Pricing for TCP*; IEEE Journal on Selected Areas in Communications, Vol. 13, No. 7, September 1995.
5. M. Feier: *Design and Prototypical Implementation of a User-based IP Accounting Module for Linux*, Diploma Thesis, University of Zürich, Switzerland, February 2007.
6. Flexible NetFlow Homepage, http://www.cisco.com/en/US/products/ps6601/products_data_sheet0900aecd804b590b.html, May 2007.
7. Iperf Homepage: http://dast.nlanr.net/Projects/Iperf/, May 2007.
8. Juniper Homepage, http://www.juniper.net/, May 2007.
9. M. Koukal, R. Bestak: *Architecture of IP Multimedia Subsystem;* 48th International Symposium ELMAR-2006 focused on Multimedia Signal Processing and Communications, Zadar, Croatia, June 2006.
10. C. de Laat, G. Gross, L. Gommans, J. Vollbrecht, D. Spence, *Generic AAA Architecture*; RFC 2903, August 2000.
11. Narus Homepage, http://www.narus.com/, May 2007.
12. B. C. Neuman, T. Ts'o: *Kerberos: An Authentication Service for Computer Networks*, IEEE Communications, Vol. 32, No. 9., pp 33—38, September 1994.
13. NIPON: *Nutzerbasiertes IP Accounting;* http://www.icsy.de/forschung/nipon/, May 2007.
14. C. Rigney, S. Willens, A. Rubens, W. Simpson, *Remote Authentication Dial In User Service (RADIUS)*; RFC2865, June 2000.
15. J. Sermersheim (ed): *Lightweight Directory Access Protocol (LDAP): The Protocol;* RFC 4511, June 2006.
16. UserIPacct Homepage: http://ramses.smeyers.be/homepage/useripacct/, May 2007.
17. UTA Dragon Homepage: http://www.crash-override.net/utadragon.html, May 2007.
18. G. Vinton (ed), *Guidelines for Internet Measurement Activities*; RFC 1262, October 1991.
19. S. Waldbusser, *Remote Network Monitoring Management Information Base*; RFC 1757, February 1995.
20. G. Zhang, B. Reuther: *A Model for User Based Traffic Accounting*; 31st EUROMICRO Conference on Software Engineering and Advanced Applications, Porto, Portugal, August 30 — September 3, 2005.
21. G. Zhang, B. Reuther, P. Mueller, *Distributed Agent Method for User Based IP Accounting*, 7th CaberNet Radicals Workshop, Bertinoro, Forlì, Italy, 13-16 October 2002.

A Protocol to Support Multi-domain Auditing of Internet-based Transport Services

Frank Eyermann
Information Systems Laboratory (IIS)
Universität der Bundeswehr München
Munich, Germany
Frank.Eyermann@unibw.de

Burkhard Stiller
Communication Systems Group CSG, IFI
University of Zürich
Zürich, Switzerland
stiller@ifi.unizh.ch

Abstract—Auditing of Service Level Agreements (SLA) defines the process of monitoring whether a service provider delivers agreed upon service levels or not. While frameworks exist to monitor application-level SLAs, the end-to-end monitoring of IP-carrying SLAs, especially in a multi-domain environment like the Internet, is still an open issue.

This work analyzes methodologies how IP-carrying SLA parameters could be efficiently measured and develops a protocol, which supports this process and minimizes overhead.

Keywords: SLA; Auditing; SLA Monitoring; Performance Metric; End-to-End Measurement

I. INTRODUCTION

An adequate access to the Internet, mainly in terms of reliability, became an Achilles' heel for many businesses. If a web-shop is not available or reacts too slowly on requests, customers buy somewhere else. But also in Business-to-Business (B2B) communications [14] it is important carrying out transactions in time. A disruption of network connectivity or even only a significant degradation of the service quality could result in a financial loss. To reduce these risks, companies utilizing the Internet as a means for commercial communications contract Service Level Agreements (SLA) [4] with Internet Service Providers (ISP). SLAs manifest those service levels the ISP will provide and the customer needs to pay for. Most valuable for customers are SLAs guaranteeing network connectivity end-to-end [9], i.e. from an origin of a connection the whole path to the destination, even in multi-domain scenarios, when the destination can be reached only through another provider as the one the origin's access is based on.

SLAs as well as all other agreements need to be monitored: Does the contractual partner fulfill its obligations as stated? This task is referred to as SLA Monitoring or Auditing [12], [7], two terms which are used in this context synonymously. Auditing can be regarded in an abstract manner as the comparison of nominal with actual values. For IP-carrying SLAs, nominal values are provided by the SLA, actual values need to be measured in the network for a given situation. In order to measure unambiguously actual performance values, metrics need to be defined, which describe the measurement setup and the relevant process in sufficient detail.

In a multi-domain scenario, like the Internet, where different network operators need to cooperate in order to provide an end-to-end connection, measurement also needs to determine where, i.e. in which domain, SLA parameters have been violated. This increases the measurement effort drastically and, therefore, makes auditing of IP transport quite "expensive". This paper presents a solution to the problem of determining a promising approach to reduce the measurement effort in a multi-domain scenario.

Section II of this paper presents related work and introduces important terms for this work. In Section III different measurement techniques are analyzed and their suitability for auditing is evaluated. Section IV presents the design of a new protocol called MeSA (Measured Signaling for Auditing), which helps to minimize the measurement effort in multi-domain environments. While Section V evaluates the approach taken, finally Section VI summarizes the key achievements and draws conclusions.

II. RELATED WORK

Related work on the problem statement given above is basically addressing three areas: (1) the description with which agreements between contracted partners can be formalized, (2) measurement approaches, and (3) auditing systems. Based on an appropriate combination of these three aspects the basis for auditing multi-domain transport services is laid.

A. Service Level Agreements

SLAs are a well known concept to manifest services and the way how these services are delivered between a service provider and a service consumer [4]: This approach is valid in various business branches, starting from housekeeping and ending in Internet-based transport service deliveries.

In general, an SLA consists of three parts [12]: The *involved parties, SLA parameters,* including metrics and algorithms used to compute them and *Service Level Objectives (SLO) and actions,* which have to be taken upon violation of SLA parameter thresholds.

The more enterprises need to depend on the Internet, the more SLAs have been applied in IT businesses as well. Most SLAs also contain bonus and penalty regulations for the case, when a provider delivers services very well or when SLOs are violated. An SLO defines a single quantity, which can be measured individually and for which the SLA defines a threshold to be met by the provider. Thus, SLAs determine an

This work has been performed partially in the framework of EMANICS (FP6-2004-IST-4-026854-NoE).

2nd Int'l Conf. on Internet Monitoring and Protection (ICIMP 2007), Silicon Valley, U.S.A., July 1—6, 2007
Best Paper Award

approach to share the risk of financial loss, when agreed upon services are not delivered as contracted.

Unfortunately, in the area of network provisioning, which is fundamental for all IT branches, currently a low dissemination of SLAs can be observed. The reason for this is that nowadays the Internet lacks the possibility to measure accurately and adequately and to report on given SLAs, thus, making agreements themselves hard to be implemented [4].

B. *IP Performance Metrics*

The IP Performance Metrics working group (IPPM WG) is a working group of the Internet Engineering Task Force (IETF). Its goal is to define metrics to be applied to quality, performance, and reliability of Internet data delivery services [10]. Additionally, the working group defined a general framework for accurately measuring and documenting metrics [21]. Metrics in this sense are carefully specified quantities related to the performance and reliability of the Internet.

Metrics are essential whenever formal and practical definitions of performance parameters are needed: For example, for an SLA containing an SLO with an upper limit for delays, it is mandatory for each partner to have the same understanding of delay and its measurement.

The IPPM WG offers neither definitions nor suggestions on *how* performance parameters are measured. Their emphasis is on definitions and the unambiguous understanding *what* a parameter expresses in order to allow measurement results to be compared, shared, and validated by different entities.

Metrics defined by the IPPM working group can be used as building blocks to compose SLOs for an SLA. In contrast to the bottom-up approach of IPPM, Section III.A analyses measurements from a top-down perspective, looking which performance metrics on the network layer influence the performance of distributed applications.

C. *Auditing*

Auditing can be generally defined as the act of assessing the validity of a process according to rules [7]. A financial audit, *e.g.*, determines if financial statements of a company are in accordance with the law.

In IT the term auditing has two meanings. A security audit is the search through a computer system for security problems and vulnerabilities [25]. This is mostly performed by inspecting in an offline manner existing system logs for suspicious entries or known attack patterns. The second meaning is the one referred to in this work: auditing as the process of checking the compliance of actual provided service quality with an SLA specification.

Related work in this area can be found in the *Web Service Level Agreement* (WSLA) framework [12]. It is a system to measure and monitor QoS parameters for application-level SLAs, especially for Web Service. A language to express SLAs and a run-time architecture to remotely monitor web services is included. Even though the authors propose a very flexible system, it is specialized for application-level SLAs and, as the end-to-end character of network SLAs is not appropriately represented, less suitable for network SLAs.

An approach for auditing of network SLAs can be found in [27]. Auditing here is an addition to an Authentication, Authorization, and Accounting framework based on the AAA standard protocol “Diameter” [2]. Not further defined “QoS equipment” is part of the service equipment and performs the actual auditing. However, the auditing process itself, i.e. the measurement of single performance parameters, is not described, so a central question remains open. An open framework for Auditing has been developed in [7].

The Distributed Accounting and Auditing for Multiple Mobile Network Operators (DAMMO, [5]) project defines auditing procedures and data exchange in close detail. A4C-Servers (Authentication, Authorization, Accounting, Auditing and Charging-Server) perform in collaboration with service equipment the actual auditing. Auditing procedures for handovers and network changes are specified. Again, the measurement and validation process is not defined. It is assumed service equipment is able to perform these tasks.

III. MEASURING OF PERFORMANCE

Once metrics and frameworks as well as mechanisms for an auditing purpose are defined, the analysis of existent possibilities on how to verify SLOs agreed upon in SLAs is essential. This happens by discussing in Subsection A, what typical SLOs in IP carrying SLAs could be and by explaining in Subsection B, how these could be measured in practice.

A. *Performance Metrics*

An SLO has been defined in Section II.A as a single quantity, which can be measured individually and for which a threshold exists. This definition of IPPM could also be applied to describe a performance metric with a threshold.

But which performance metrics can be found in SLAs and with which thresholds? The answer needs to be derived from traffic profiles of typical Internet applications. Application categories being of interest for auditing include:

- Real-time applications (*e.g.*, Voice-over-IP)
- Multimedia applications (*e.g.*, Video streaming)
- Bulk-data transfer (*e.g.*, GridFTP download)
- Client-Server applications (*e.g.*, distributed simulation)

Analyzing these applications the following transport layer performance parameters can be identified [6]:

- Throughput (i.e. available Bandwidth)
- Per-flow sequence preservation
- Packet loss
- One-way delay
- Round-trip time
- Jitter

Availability is no transport layer performance parameter. From a flow-level point of view a hardware failure is seen as complete packet loss over a longer interval. Still, availability is stated in many SLA as a time share (*e.g.*, 99.9%) at which the system is available to provide services. It needs to be monitored, even if no application is sending any data.

B. Measuring Performance Metrics

After having identified relevant performance parameters the next step is to analyze how they can be measured in order to monitor SLOs. Different methodologies to measure parameters exist, but none of which is universal in the sense that it is useful for each performance parameter. The most common measurement methodologies include:

- Passive measurements monitor only packets and evaluate information in packet headers and signaling information. For the measurement itself no additional network traffic is generated, but during the measurement huge amounts of data may accrue, which have to be transferred afterwards to another host for off-line evaluation and correlation.
- Brokers keep an account on vacant resources in the network and allocate them to services.
- Counters and gauges are maintained internally by routers and other network equipment in order to provide information on their status and the status of the network (*e.g.*, as part of Management Information Base, MIB).
- Probing is the injection of artificial packets (so called probes) in the network. Special measurement agents are placed in the network, which monitor how probes are treated by the network. Only probes are metered.
- Inference techniques, sometimes referred to as network tomography, are a subclass of probing techniques, for which measurement agents are only deployed at the source, mostly combined in a single program with the probe generator. This technique uses standard behavior of protocol stack implementations on nodes in the network to receive a feedback to requests the program sent. The most widely know tool using such a technique is traceroute, which sends probes with increasing values in the TTL-field of the IP header in order to achieve information about intermediate systems on the way to a destination host, from time-to-live exceeded ICMP-messages returned by routers.

C. Applicability of Measurement Methods to Performance Parameters

These measurement techniques presented in Section III.B have different strengths and weaknesses when used to measure the performance parameters listed in Section III.A. In order to reduce the measurement overhead and improve the measurement accuracy, it is beneficial to choose always the most appropriate measurement technique.

This “appropriateness” is shown in Table I. Within this table all different methods are rated in 6 categories:

- Well suited (++): measurement is possible with good accuracy and less effort.
- Suited (+): measurement is possible, but accuracy is lower and/or effort is higher than for well suited ones.
- Possible (0): measurement is possible, but accuracy is low and/or effort is significant.
- Difficult/very difficult (-/--): measurement is theoretically possible, but not economically achievable. The effort necessary is much higher than the gain, especially as there are better suited methods.
- Not possible (X): measurement is not possible, even with a high effort.

TABLE I. COMPARISON OF MEASUREMENT METHODOLOGIES FOR PERFORMANCE PARAMETERS

	Connectivity	RTT	Throughput	Sequence	Loss	Delay	Jitter
Passive	0	-	++	++	++	-	-
Broker	X	X	+	X	X	+	-
Counters Gauges	0	X	0	X	0	0	-
Probing	++	++	-	X	+	++	+
Inference	++	++	-	X	+[1]	+	-

++ = well suited, + = suited, 0 = possible,
- = difficult, -- = very difficult, X = not possible

In a more detailed discussion on TABLE I. another criteria has to be included. Each performance parameter has special requirements, where in the data path measurement has to be performed. Four different locations can be identified:

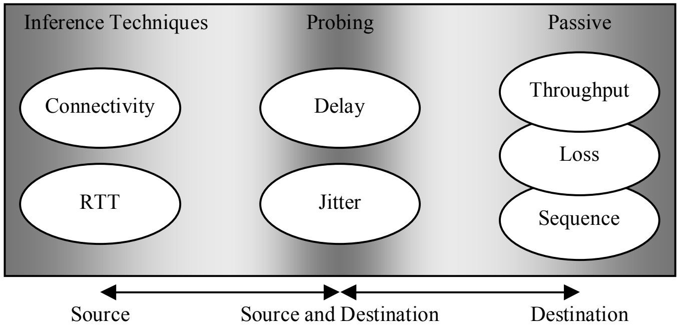
- Anywhere in the data path
- Only at the source
- Only at the destination
- At source and destination

Between efficient measurement technologies and measurement locations correlations can be found:

- A passive measurement is well suited if only a measurement at the destination is necessary (*e.g.*, loss and sequence). Information from the source needs to be communicated to the destination or a central measurement coordination station, which raises the effort drastically (*e.g.*, delay and jitter).
- Brokers show typically only a general overview on the network. They can only warn, when a state of the network is reached, where load-dependent performance parameters might degrade (*e.g.*, throughput and delay). When congestion in the network is detected, brokers can not measure the impact in detail.
- Counters and gauges can provide information on the state of routers. Quite similar to brokers, counters and gauges can warn, if a state is reached threading network performance. Thus, they allow only monitoring the networks health status, but can not provide detailed values for performance parameters.
- Probing has its strength, if the actual traffic does not need to be looked at, because the measurement is only performed with these probes. Probes can be created in an application-specific manner, which allows for measuring at different places in the network, i.e. at the source and the destination.
- Inference techniques are a subclass of probing techniques, and they show quite similar strengths and weaknesses. But since no measurement infrastructure is in place, probes have to be reflected to sources, which might cause additional distortion. Therefore, this technique is best suited for performance parameters, which can be measured at the source.

[1] Values might be quite inaccurate.

Figure 1. Relationship of measurement methodologies and performance parameters



In summary, Fig. 1 shows these relationships. Inference techniques deployed at the source are best suited for measuring connectivity and round-trip time, while probing with measurement agents at least on the destination can be used for measuring delay and jitter. Passive techniques are best deployed on the destination and can measure loss and per flow sequence preservation.

D. Auditing in Multi-domain Scenarios

Based on the discussion above, it can be seen that it is beneficial for auditing to use different measurement methods at different locations in the data path in order to work efficiently. This has considerable consequences for auditing in a scenario, where more than one ISP has to be traversed between a sender and a receiver. This is often referred to as the multi-domain characteristic [5].

Fig. 2 shows an example, where a sender is connected over three different networks to a receiver. It is assumed that each network represents one domain. On one hand, these ISPs need to work together in order to transfer data from a sender to a receiver, on the other hand, they determine independent business entities with their own goals and policies. Recent history has shown that ISPs are not willing to accept restrictions in the autonomy over their networks [13].

Multi-domain auditing now requires that each single domain fulfills its SLA, which has to be examined, too. This means, that within each domain all relevant performance parameters have to be metered. This is displayed in Fig. 2. Throughput, round-trip time, connectivity loss, and per-flow sequence preservation is measured (1) at the appropriate places for the whole scenario, i.e. source and destination which are sender and receiver respectively, and (2) within each domain, i.e. source is the ingress router. The destination is the egress router of the domain. The auditing overhead, therefore, increases drastically. In this scenario delay for examples has to be measured four times, i.e. within each domain and end-to-end. The MeSA protocol, developed and described in the following section, reduces this overhead to a minimum, but still complies with multi-domain requirements and ISP autonomy.

IV. MeSA Protocol Design

As seen, a significant effort has to be made performing auditing. Especially the multi-domain characteristic, requiring measurement within each domain, boosts complexity. A novel protocol termed MeSA (Measured Signaling for Auditing) defined here can reduce this effort by combining measurement and signaling. Before details of this protocol are presented in Subsection 0E and the following, major design decisions are discussed.

A. Measurement Scope

Most of today's professional routers are able to monitor path conditions (*e.g.*, Active Traffic Monitoring implemented in Cisco IOS, [26]). It is possible to program the devices to monitor loss and round-trip time between itself and a preconfigured destination device. The device will periodically send probes to the destination device, which returns these packets.

With the help of this feature an operator can monitor all links in the network and create statistics, which can be used to generate a detailed picture of major conditions in a network. If all paths in the network are monitored, the picture is complete and for any flow in the network it is theoretically possible to approximate the flow's performance parameters.

The key advantage of this method is that the monitoring of links is already implemented in existing network hardware; no change or update of hardware is required. Another privilege is that the overhead caused by this type of measurement is foreseeable: As each link needs to be monitored, the overhead is a function of the number of links, which typically remain very constant.

But there are also drawbacks to be aware of. For link-based monitoring, monitoring parameters are chosen per link. This prevents an application- or user-specific monitoring, where monitoring parameters are chosen to fit the characteristics of an application or the needs of a user.

In order to calculate end-to-end performance parameters, it is necessary to know which path the packets took through the network. This is a non-trivial task. Link failure, dynamic

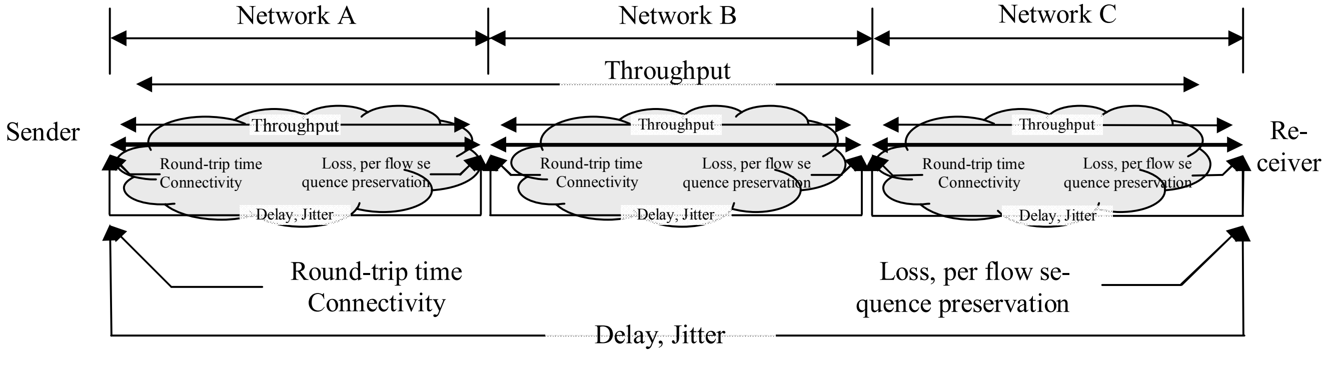


Figure 2. Measurement for multi-domain auditing

routing, or load-balancers might lead to non-deterministic routing decisions requiring to asset the concrete path on a regular basis. The Trial-Balloon Procedure [5] defines one mechanism, which is able to determine the path of a flow.

Another possible approach is a flow-based measurement. For the notion of a "flow" several definitions exists. A very broad definition by the IP Flow Information Exchange (IPFIX) working group can be found in [22], which groups packets according to a set of common header fields. While this definition is valid from the perspective of IPFIX, for auditing it is too coarse-grained. Here a flow is defined as an application-level end-to-end stream, where all packets exhibit a common traffic profile. The significant difference of both definitions is that the latter does not allow for aggregation and it is independent of any fields of the IP header.

With a flow-based measurement, probes are included in a user's flow with the same address information. They, therefore, take the same route through the network as the user data, independently which one it is. The problem finding the correct path is solved by the network itself.

As another advantage user and application requirements can be taken into account when generating probes, *e.g.,* during a Voice-over-IP session with silence suppression, no or less probes are generated during periods of silence.

A disadvantage of this approach is that the auditing overhead is not foreseeable for a provider. For flow-based measurements the number of probes and, therefore, the overhead is a function of the number of flows under audit. This number is highly dynamic and difficult to predict. Furthermore, especially in backbone networks, there will be more than one flow with a turned on auditing function on one link. This doubles the work as the information about link conditions is already determined by one set of probes.

Weighting drawbacks and advantages of both approaches presented, shows that the management of a flow-based measurement is simpler and application- and user-specific requirements can be taken into account. Discovering the path of a flow is almost as costly as the measurement itself, taking into account that the path discovery has to happen on a regular basis in order to spot path changes quickly.

But the most significant advantage of a flow-based measurement approach is the possibility to shift load from the network to end-hosts, helping to improve scalability [23]. This topic will be discussed in more detail in the following section.

B. Management of Measurement Data

Within a measurement scenario, two general types of data can be observed, which are caused by measurement: signaling and probing traffic. Signaling traffic contains information, which controls the measurement process, i.e. starting and stopping of sessions, or negotiations about measurement parameters. The control part of the *One-Way Active Measurement Protocol* (OWAMP, [24]) termed OWAMP-Control is an example for such signaling traffic.

The second type is probing traffic. As the name already suggests, it consists of probes used when probing techniques are applied. In general this is a series of packets, where the type of packets, packet size, packet frequency, and duration of the series depends on the measurement algorithm and most probably on parameters negotiated. The payload of these packets is mostly without any meaning. Some algorithms include a sequence number, a timestamp, or addressing information in the payload, but these occupy only a fraction of the space available. Any remaining space is filled with a random sequence of bytes, in order to impede potential compression.

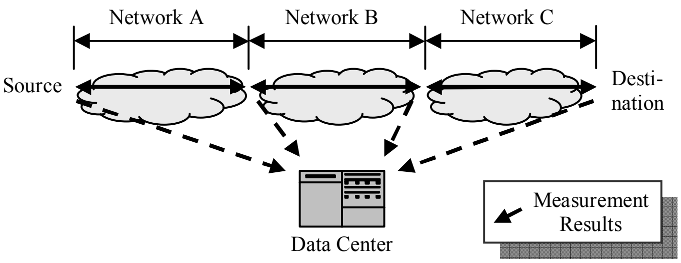


Figure 3. Measurement data transfer

If those results measured need to be processed further or archived, they must be transferred as signaling packets to data centers. This may produce a high overhead in the network and a bottleneck on the data center, because a distribution of this load on different data centers is often not possible (*e.g.*, one center may need to hold all information on one flow in order to calculate all delays required).

In this work, the separation of probing and signaling traffic is called *out-of-band measurement*[2]. The name "out-of-band" is chosen, because the stream of data contained in these probes can be seen as one "band". The stream of signaling data is not contained in this stream, therefore, it is "out-of-band".

In order to avoid the disadvantages mentioned above *in-band measurement* is proposed. Following the definition from the last section, in-band measurement would mean, carrying all information in one single band, i.e. in the payload of probes.

This avoids additional signaling traffic, thus, it reduces the overhead significantly. This is even more important for auditing. As illustrated in Fig. 3 and in order to locate violations, intermediate results from borders of each administrative domain need to be transferred to a data center for further evaluation. "Data center" here stands for the role of a system, which evaluates measurement results. This role may be taken up also be several systems, which share the evaluation process.

On the first glance, sending signaling information within the data stream seems to be unreasonable. As it can be seen in Fig. 3, the paths of signaling traffic and probing traffic are different; measurement results are reported to the data center and probing traffic is transferred to the destination host. Sending measurement information in the payload of data packets would mean that all measurement results will arrive at the destination only.

The remainder of this paper develops the protocol which makes use of in-band measurement to support auditing. The protocol is designed to work in multi-domain environments, and supports passive, but especially probing measurements.

[2] The term "out-of-band" and the its opposite "in-band" can be found in the measurement context with different meaning, too (see, *e.g.*, [1]). If Quality-of-Service (QoS) mechanisms are in place, out-of-band measurement can mean sending probes within a different traffic class; in-band means sending probes within the same traffic class valid for the data themselves.

C. *Principle Operation and Characteristics*

MeSA uses probing to measure delay and jitter. A sender periodically intersperses MeSA probes in the application packet stream, i.e. a flow as defined in Subsection A. Each *MeSA-enabled router* (or simply *MeSA router*) appends a time-stamp to these probes, whenever the packet is received, before queueing the packet for sending. The receiver can calculate the delay between each two MeSA routers from these timestamps. Not all routers need to be MeSA-enabled. In case of auditing, only per-domain results are of interest. Therefore, only at borders of these domains MeSA routers need to be established. Thus, the ISP autonomy of a domain to be traversed has been achieved.

Fig. 4 shows a sample scenario, where two end-systems are connected via two providers (ISP A and ISP B). These ISPs run operational MeSA routers at their borders. Besides "normal" application data and their respective packets, the sender periodically generates MeSA probes. MeSA routers append timestamps to these probes and, finally, the receiver evaluates these probes and calculates delays. In the lower part of the graphic it is shown, how one probe evolves over time. The probe grows with each MeSA router it traverses by one entry[3]. The time the probe spent in each domain is calculated by subtracting the timestamps.

This approach is tailor for auditing, finding a tradeoff between measurement error and effort. Errors could occur, because processing delays in routers are not considered and the synchronization of the clocks of MeSA routers might be imprecise, especially, when synchronization is performed over the network. The impact of the first issue is tolerable, since the processing delay is several magnitudes smaller than typical SLO thresholds, some 10 µs versus some 10 ms.

Concerning the second issue, the Network Time Protocol (NTP) is designed to synchronize system clocks to the accurate time as close as possible. The NTP software is available for almost every workstation and server operating system, and it is implemented in nearly every router and switch. The Network Time Protocol Version 3 [15] has provides nominal accuracy in the lower range of milliseconds, The new NTP Version 4 [16] is expected to improve accuracy by a factor of ten [17], thus, the synchronization error is also tolerable.

D. *Phase 1 "Negotiation"*

An auditing session starts with the negotiation phase. Its task is, firstly, to authorize the user for requesting and running auditing services, and secondly, to supply all parties with necessary information. The overhead introduced by this phase has to be minimal; especially a long latency time before the data transfer can start has to be avoided.

Authorization is done by edge routers, to which users are connected to. On requesting a new auditing session, routers forward the request to an authorization server, which looks up the user's profile and determines the user's authorization to use auditing. In case of a failure the server returns an error message, in case of success a session identifier (SID). The SID is unique for the forwarding edge router and for the lifetime of a session. The edge router stores the SID in an authorization table and sends it to the user. The user is supposed to include the SID in every MeSA probe, and the edge router will check, if MeSA probes include a valid SID.

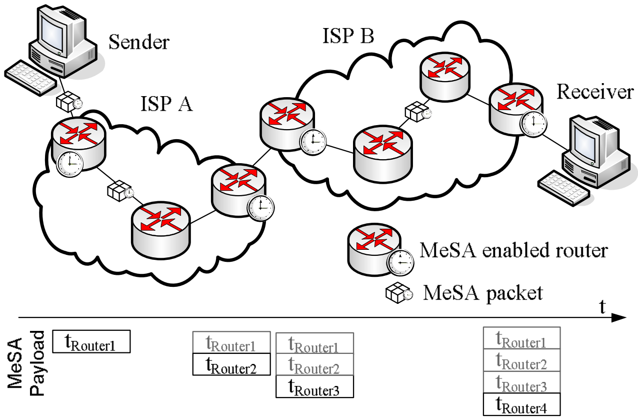


Figure 4. MeSA principle operation

The MeSA protocol stores state information only in the sender, the receiver, and, if required, in the edge router of an ISP, the sender is connected to. No state is stored in intermediate systems in order to avoid scalability problems.

E. *Phase 2 "Data Transfer"*

During the lifetime of a flow, which shall be audited, the sender periodically intersperses MeSA probes in the data stream. MeSA probes are IP packets, which share the same IP header as the data packets- with two exceptions:

1) MeSA probes set the Router Alert Option. The Router Alert Option is defined in RFC 2113 [11] and RFC 2711 [20] for IPv4 and IPv6, respectively, and enable the notification of intermediate routers that a packet needs to be analyzed and processed before being forwarded. This option permits a MeSA router to quickly identify MeSA probes in the packet stream by only processing the IP header fields.

2) For IPv6 the probes must have a different flow label, as required by RFC 2460 [3].

A MeSA probe consists of

- an IP-Header with Router Alert Option (see above)
- a *MeSA Header*
- *MeSA Entries* from each MeSA router traversed.

The MeSA header is depicted in Fig. 5. The first four bits show the MeSA protocol version (Ver), followed by eight bits which are reserved for future use (Res). The SID (Session Identifier) field holds the SID in terms of 16 bits, assigned by the ISP during the negotiation phase. The SeqNo (Sequence Number) field with 16 bits is incremented by one for each MeSA probe sent and allows for the detection of the loss of probes. In order to link MeSA probes to the application data, each MeSA probe holds the flow label and the sequence number of the last data packet sent in the Flow Label field and the Appl SeqNo (Application Sequence Number) field, respectively. This sequence number, in terms of 32 bits, has to

[3] An IP packet with 1500 bytes can hold up to 48 entries with 30 bytes from MeSA routers. Given that an Internet packet on average only traverses approx. 16 routers at all [19], this should be sufficient for any healthy Internet path.

be provided by the transport-layer protocol the application data is transferred with (*e.g.*, TCP or RTP).

0	Ver	Res	Flow Label	SID	SeqNo
8	Appl SeqNo				
12					

Figure 5. MeSA protocol header

The format of a MeSA Entry appended by each MeSA router traversed is shown in Fig. 6. In the current stage Bit 0 of the Flag field F shows if a signature is present. The rest of seven flag bits are unused. Furthermore, 8 bits of the reserved field (Res) are reserved for future use.

The ASN is the Autonomous System Number of the operator, which is assigned by the Internet Assigned Numbers Authority (IANA) [8]. The RID (Router Identifier) identifies one router within one Autonomous System. The assignment of the RID number is performed by the operator and does not need to be public. Again, the ISP autonomy is achieved. In the timestamp field (8 bytes) the router stores its current time as milliseconds past midnight 1970/1/1, a format which is usual on many Unix systems. Each entry can be signed optionally, in order to detect malicious manipulations. Thus, a 16 byte Signature field has been integrated.

n+0	F	Res	ASN	RID
n+6	Timestamp (8 Byte)			
n+14	Signature (16 Byte) (optional)			
n+22				
n+30				

Figure 6. MeSA entry

The probes included by the sender during the data transfer phase are evaluated at the receiver. The timestamps in the probes, and the order and time of arrival of the probes enable the receiver to calculate one-way delay and jitter. Probes which show SLA violations are stored for later complaint at the operator. Throughput, loss and per-flow sequence preservation are measured in a passive manner by the receiver.

F. *Phase 3 "Tear down"*

After the data transfer is completed, the session needs to be torn down. Besides the release of all state information, the auditing results are transferred to the initiator of the session. Only the initiator of an auditing session, as the one who was authenticated and authorized, can complain SLA violations at his operator. Whether the sender or the receiver is in the role of the initiator depends on the traffic profile of the used application. In case of the receiver being the initiator, no data needs to transferred, as the data is already with the initiator. In the other case the auditing results are sent back to the sender.

Each ISP operates a complaint desk, where SLA violations can be reported to. The complaint is sent automatically, including the SID and the probes, proofing the violation.

V. EVALUATION

The MeSA protocol and router extensions have been evaluated by simulation and a first prototypical implementation.

A. *Simulation Results*

First simulations have been performed using ns-2 [18]. A scenario has been setup with 18 MeSA enabled border routers and 10 domain-internal routers in 6 domains. Fig. 7 shows the scenario without detailing all those routers. Twelve systems are placed in the scenario generating background traffic, which is not measured. The end-system labeled "Start" is used as a sender and the end-systems a, b, and c are used as receivers for MeSA flows.

The MeSA protocol was not implemented as an agent for ns-2, instead the protocol's algorithms were applied to trace files generated by the simulator. This has the advantage, that runs with different parameters can be performed on the same data easily.

Simulations have been performed to determine the impact of different sampling periods on measurement accuracy. Further simulations are used to evaluate algorithms to eliminate clock synchronization errors ex post. With the low fraction of non MeSA-routers in our scenario results have been promising.

B. *Prototypical Implementation*

An implementation architecture for the MeSA protocol is shown in Fig. 8. It consists of several modules, deployed on respective systems: sender, edge router, MeSA router and receiver. In this figure the sender is the initiator. For the receiver being the initiator the edge router has to be deployed in front of the receiver.

The *Initiator* modules on the sender and receiver establish new auditing sessions. The *Access Controller* on the edge router authenticates and authorizes the request. MeSA probes are generated by the *Probe Generator*. This can happen periodically or it may be triggered by the application, if the application features auditing support. The edge router checks each probe for is validity and the *Probe Stamper* appends a timestamp to each probe. On the receiver the *Auditor* evaluates probes as described in Section IV.E. The *Complaint Handler* transfers auditing results to the initiator and complains SLA violations at the ISP's complaint desk (not shown).

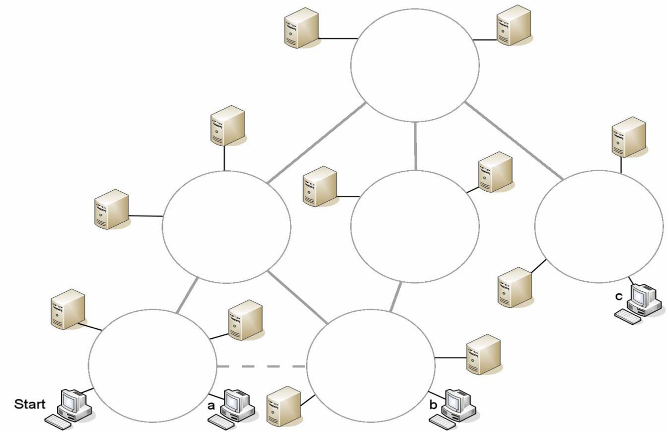


Figure 7. Simulation scenario

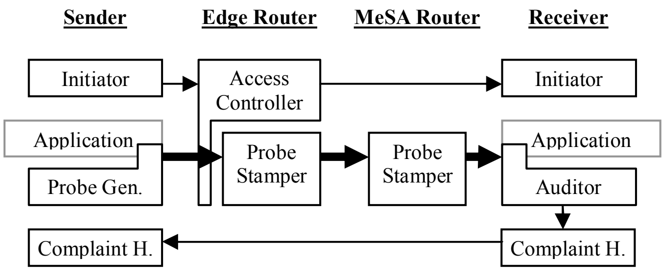


Figure 8. Auditing System Software Architecture

The prototype already features the Probe Generator module on the sender and the Probe Stamper module on the routers. A basic Auditor module on the receiver filters the probes out of the data stream and stores them for offline processing.

For the prototypical implementation, Linux systems have been used as sender, receiver, and routers. IPv6 was chosen for the application and probing traffic. We identified the implementation of the Router Alert Option as a major implementation problem. Reasons are minimal and wrong documentation respectively[4].

As another obstacle, the Linux developers allow the Router Alert option only for raw sockets. This undocumented fact requires manual processing of higher level protocol on routers.

VI. SUMMARY AND OUTLOOK

This paper presented a protocol in support of auditing, especially of IP-carrying SLAs. As a first step different measure methods have been analyzed and evaluated for their appropriateness to measure performance parameters of SLAs. A concise matrix has been sketched, showing the relationship of measurement method, performance parameter and measurement location. Hereupon the new MeSA protocol (Measured Signaling for Auditing) has been designed, combining the most suitable mechanisms. Simulation results and a first prototypical implementation have been described.

Because of limited space in this paper security related aspects of the MeSA protocol are not discussed. However, measures have been defined in order to authenticate and authorize auditing requests, to ensure the authentication, integrity and confidentiality of the data transferred in the negotiation and tear-down phase, as well as to ensure the authentication and integrity of the data in the MeSA probes. Due to the distributed nature of the MeSA protocol the last issue requires that each router has to sign its MeSA entry.

In the next phase of this work more detailed simulations will be undertaken with the help of topology generators. This will raise significance of those initial results achieved.

REFERENCES

[1] Breslau, L.; Knightly, E.W.; Shenker, S.; Stoica, I. & Zhang, H., Endpoint admission control: architectural issues and performance, SIGCOMM '00: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, 2000, ACM Press, New York, NY, USA, pp. 57--69.

[2] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., Arkko, J.; “Diameter Base Protocol”, RFC 3588, Sept 2003

[3] Deering, S., Hinden, R.; “Internet Protocol, Version 6 (IPv6) Specification”, RFC 2460, Dec 1998

[4] Engel, F.; “The role of service level agreements in the internet service provider industry”, Int. J. Netw. Manag., John Wiley & Sons, Inc., 1999, 9, 299-301

[5] Eyermann, F., Racz, P., Schaefer, C., Stiller, B., Walter, T.; „Distributed Accounting and Auditing for Multiple Mobile Network Operators: Architecture” DoCoMo Euro-Labs Internal Technical Report I-ST-012, April 2005.

[6] Filsfils, C., Evans, J.; Engineering a multiservice IP backbone to support tight SLAs, Computer Networks, Volume 40, Issue 1 , September 2002, p. 131-148

[7] Hasan, H., Stiller, B.; A Generic Model and Architecture for Automated Auditing, 16th IFIP/IEEE International Workshop on Distributed Systems: Operation and Management (DSOM 2005), Barcelona, Catalunya, Spain, October 24-26, 2005, pp 121-129

[8] Hawkinson, J., Bates, T.; “Guidelines for creation, selection, and registration of an Autonomous System (AS)”, RFC 1930, March 1996

[9] Ho, K., Howarth, M., Wang, N., Pavlou, G. and Georgoulas, S.; Two approaches to Internet traffic engineering for end-to-end quality of service provisioning, Next Generation Internet Networks, April 2005, pp 135- 142

[10] IETF IP Performance Metrics working group, http://www.ietf.org/html.charters/ippm-charter.html

[11] Katz, D.; “IP Router Alert Option”, RFC 2113, Feb. 1997

[12] Keller, A., Ludwig, H.; “The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services“, Technical report RC22456, IBM Research.

[13] Mahajan, R.; Practical and Efficient Internet Routing with Competing Interests, Ph.D. Dissertation, University of Washington, Dec. 2005

[14] McGregor, C., Kumaran, S.; “Business process monitoring using web services in B2B e-commerce”, Parallel and Distributed Processing Symposium. (IPDPS 2002), Fort Lauderdale, Florida, USA, April 15-19, 2002, .pp 219-226

[15] Mills, D.; “Network Time Protocol (Version 3) Specification, Implementation and Analysis”, RFC 1305, March 1992

[16] Mills, D., Plonka, D., Montgomery, J.; “Simple network time protocol (SNTP) version 4 for IPv4, IPv6 and OSI”, RFC 4330, Dec 2005

[17] Mills, D.; “Network Time Protocol (NTP) General Overview”, University of Delaware, http://www.cis.udel.edu/~mills/ntp.html, visited July 2006

[18] ns-2 project homepage, http://nsnam.isi.edu/nsnam/index.php/Main_Page, visited Feb 2007

[19] Packet wingspan distribution, http://www.nlanr.net/NA/Learn/wingspan.html, visited Feb. 2007

[20] Partridge, C., Jackson, A.; “IPv6 Router Alert Option”, RFC 2711, Oct. 1999.

[21] Paxson, V., Almes, G., Mahdavi, J., Mathis, M.; “Framework for IP Performance Metrics”, RFC 2330, May 1998

[22] Quittek, J., Zseby, T., Claise, B., Zander, S.; “Requirements for IP Flow Information Export (IPFIX)”, RFC 3917, October 2004.

[23] Saltzer, J., Reed, D., Clark, D.; “End-To-End Arguments in System Design”, ACM Transactions on Computer Systems 2(4), 277-288, 1984

[24] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., Zekauskas, M.; “A One-way Active Measurement Protocol (OWAMP)”, work in progress, Feb 2006

[25] Texas State Library and Archive Commision, Glossery; http://www.tsl.state.tx.us/ld/pubs/compsecurity/glossary.html, visited Feb 2007

[26] Tychon, Emmanuel; Advanced Performance Measurement with Cisco IOS IP SLA, Cisco Systems, http://www.etychon.com/presentations/Advanced_IPSLA.pdf, visited Feb 2007

[27] Zseby, T., Zander, S., Carle, G.; “Policy-Based Accounting”, RFC 3334, October 2002

[4] The documentation of the Linux C API describes a wrong prototype for the creation of sockets which handle packets with the router alert option set on the router.